Department of Computing

Indexing and Retrieval of Free-form Surfaces using Self-Organizing Maps

Alexander D. MacLennan

This thesis is presented for the Degree of
Doctor of Philosophy
of
Curtin University

August 2013

Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgement has been made. This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signature : .....

Date : .....

Abstract

This thesis describes the use of Self-Organizing Maps (SOMs) in combination with unique curvature based feature combinations and various training data sets as a clustering function to describe free-form surfaces. Free-form surfaces encoded as triangulated meshes were the focus of this investigation as many surface types can be reduced to this format. This SOM-based descriptor was successfully used as the basis of techniques that segmented, indexed and retrieved free-form surfaces represented as triangulated meshes by labelling the vertices of the mesh.

Our per-vertex descriptors were connected to other vertices that shared the same cluster to successfully segment free-form surfaces into components when processing denoised 3D models reconstructed from 2.5D range data and a CAD model into parts that aligned with the underlying NURBS model used as the source model of the triangulated mesh. We found that for the purposes of segmenting free-form surfaces the surface being segmented is the best choice for training data and when the mean distance between each vertex and its closest matching neuron in the SOM (cluster) is very low the surface will be over-segmented.

We used the clustering function of the SOM combined with the area surrounding each vertex to create a new ”bag-of-words” descriptor encoded as a histogram and applied this to indexing and retrieving of parts of free-form surfaces using a spherical region selection widget acquired from single 2.5D range data scans, 3D models constructed from multiple 2.5D range scans and solid CAD models. The Earth Movers Distance (EMD) metric was combined with SQL queries to reduce the search space when seeking matches within our index. Our technique worked well for low-noise surfaces and worked best when searching for self-similarity within a single surface. We applied our technique to object class recognition using a CAD model benchmark dataset and achieved precision and recall results comparable to other modern shape descriptors.

Acknowledgements

The marathon of the investigation and writing required to complete this thesis would have been impossible without the guidance, ideas, reviewing, feedback and constant encouragement from my Supervisor Professor Geoff West.

Thank you to my co-supervisor Professor Michael Cardew-Hall whom I was lucky enough to spend time discussing and working through possible approaches to the problems tacked in this thesis with the impact of these ideas is spread throughout the body of work.

Thank you to both Professor Geoff West and Professor Michael Cardew-Hall for the opportunity to pursue a PhD with a funded scholarship via the ARC Discovery Projet Grant (ID:DP0452731). I appreciate the understanding of the Department of Computing in accommodating my leave of absence due to my work commitments.

To my lovely wife Reny, an infinite source of patience, motivation and support throughout my time writing this thesis, you have your husband back.

Thank you to the thesis examiners for spending the time to evaluate my work in detail. Their thoughtful and thorough feedback greatly improved both the structure and content of this work.

Thank you to Dr. Bernard Rolfe for the cross-member CAD model used throughout the thesis.

Publications Related to this Thesis

- MacLennan, G. A. W. West, and M. Cardew-Hall. Investigating the segmentation of free form triangulated surfaces using a self-organising map. Proceedings of DETC 2006, ASME 2006 Design Engineering Technical Conferences and Information in Engineering Conference, Philadelphia, USA, September 2006. ASME.
- MacLennan, A.D. and West, G.A.W., Region Matching in Free-form Surfaces using Self Organizing Maps, Digital Image Computing: Techniques and Applications (DICTA 2008), December 1-3 2008, Canberra, ACT, Australia

Acronyms used in this Thesis

- BMU - Best Matching Unit. The neuron that contains the weights that are most similar to the feature vector being evaluated.
- BoW - Bag of Words. A pattern matching technique that relies on statistical representations of features without any structure encoded in them.
- C - Curvedness. A curvature metric that descibes the magnitude of curvature and is decoupled from the type of curvature.
- EMD - Earth Movers Distance. A distance metric that calculates the distance between histograms as a maximal flow network.
- H - Mean curvature. The mean principal curvature.
- IGES - Initial Graphics Exchange Specification. A portable file format used to encode computer aided design models.
- K - Gaussian curvature.
- k_1 - The maximum principal curvature.
- k_2 - The minimum principal curvature.
- NURBS - Non-uniform rational B-spline. A mathematical encoding used to represent surfaces.
- OOGL - Object oriented graphics library. The base libraries and file formats used in the Geomview geometry manipulation and visualisation package.
- Q - Quantization error. The distance between the feature vertices being evaluated by a SOM and it's BMU.
- S - Shape Index. A curvature metric that describes how a surface bends at a given point and is decoupled from the magnitude of that curvature.
- SOM - Self-Organizing Map. An unuspervised learning technique used to cluster and project multidimensional data into a 2D network of neurons.

- STEP - Standard for the Exchange of Product model data (ISO 10303). A standard file format used to encode CAD models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the increasing use of CAD models for manufacturing, cost factors driving the re-use of design of products, affordability of 3D model acquisition devices and home grown solutions [Microsoft Corporation, 2011], 3D models and free-form surfaces via the internet, the demand for breaking down these models into meaningful parts and the matching of common attributes across multiple parts has increased. We are motived by the problem of reusing die designs used to form sheet metal into parts by pressing sheet metal such as car doors and bonnets to reduce the consumption of resources, time and money. This is an interesting problem because it is not simple to identify similarities and differences between the free-form regions that are not separated by edges or planes. Experimentation with manufacturing dies to create parts is expensive and the use of computationally intensive simulation can be avoided or made cheaper by starting with a design that already creates a part similar to the desired output. Die designs can be reused on a global or local scale with all or sections from existing dies being used to accelerate or eliminate the prototyping process. We investigated the general problems of describing, segmenting, indexing and the retrieval of parts of free-form surfaces to identify new techniques for working toward a solution for identifying die designs that could be reused. We pose the problem of describing free-form surfaces instead of 3D models so we can apply our ideas to a wide range of source data such as triangulated meshes of 2.5D range data.

Curvature is an intuitive way to describe the properties of surfaces as it is simple to observe and describe mathematically and visually. Curvature has been used in previous research to describe surfaces with fixed combinations of these properties [Besl and Jain, 1988, Emanuele Trucco, 1995, Bose et al., 1996, Vieira and Shimada, 2005, Alrashdan et al., 2000, Yokoya and Levine, 1989, Koh et al., 1993, 1995, Bhandarker et al., 1997, Liu and Wang, 1999]. We used Self-Organizing Maps (SOMs) with multiple point-based curvature measurements, including combinations of curvature measures that had not yet been evaluated, to cluster vertices based upon these properties. The SOM is an unsupervised learning technique that uses a network of neurons directly connected to their neighbours and changes the neuron values to more closely match the input data. SOMs have been applied to a wide range of problems including character recognition, compression and segmentation [Abidi et al., 1994, Barbalho et al., 2001, Yao et al., 2000, Aksela

et al., 2003]. Previous investigations have used fixed sets of curvature properties to segment 2.5D range data [Koh et al., 1993, 1995, Bhandarker et al., 1997] using SOMs. We investigated new feature combinations, different types of training data and variations in the training parameters, to describe surfaces represented as triangulated meshes. Training data sets, feature combinations and training epoch parameters were evaluated as a part of this investigation. We chose an unsupervised learning technique over supervised and reinforcement learning methodologies because we wanted to automate as much of the learning process as possible with no intervention from a human critic beyond the evaluation of the results of our experiments or any preconceived notions of how different combinations of curvature metrics should be categorised. Using an unsupervised technique meant we could efficiently process very large training data sets without being bound by time constraints introduced by human interaction.

We applied this technique to the segmentation of many different surface types and found that using the surface being segmented as the training data set was the most reliable option for low-noise input data. For CAD models the boundaries between regions were similar to the underlying NURBS based CAD model description. The most reliable curvature metric combinations were identified and the impact of the training epoch parameter was also evaluated. It was found that for the purposes of indexing and segmenting surfaces a low mean error between a vertices' features and the cluster that best represented that error (Q) wasn't a good indicator of how well a SOM would perform in these scenarios.

A new bag of words [Blei et al., 2002, IEE, 2009] style of descriptor to describe regions on free-form surfaces was created and evaluated. Our implementation of the descriptor is a breakdown of what proportion of the surface is represented by each neuron (cluster) of a SOM and is encoded as a histogram. The effectiveness of our descriptor and its parameters are evaluated when searching single surfaces, sets of surfaces and noisy range data. We used a SQL query to reduce the search space by retrieving a subset of results based on the dominant features of the histogram describing our search criteria. We will evaluate the effectiveness of our descriptor when matching both whole surfaces and regions on a

surface.

The Earth Movers Distance (EMD) distance metric [Peleg et al., 1989, Rubner et al., 1998] was used to find matches within the subset of results. This was an appropriate choice as it was able account for similarity in terms of feature space (the Euclidean distance between neurons) when comparing histograms, a property not shared by the least mean squared difference (LMS).

In Chapter 2 we will review the literature and concepts core to our investigation. We describe the concepts of artificial neural networks at a high level and then examine the SOM in detail. The vector weights that are used to represent a neuron in the SOM and the training algorithm are detailed followed by neural topology and connectivity models. Examples of SOM visualisation techniques are then demonstrated using sample datasets. Curvature metrics are then introduced, we detail the methods for calculating the base descriptors k_1, k_2, H and K for discrete surfaces, discuss the curvature metrics C and S and show how these descriptors can be combined to identify surface types. The current state of the art free-form surface segmentation techniques for both 2.5D range data and 3D models are discussed followed by 3D model indexing and retrieval.

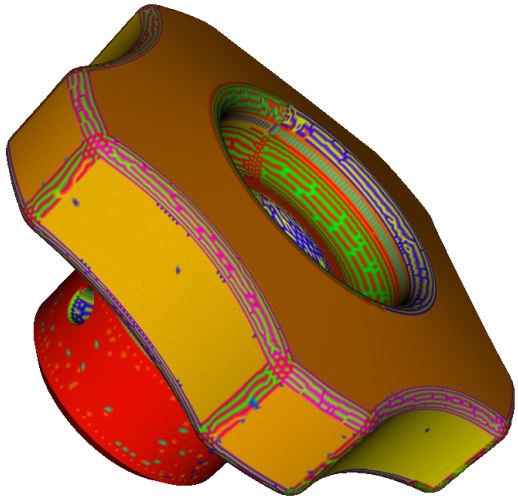
In Chapter 3 we investigate the use of a SOM with curvature metrics as input as a vertex classifier. We evaluate the factors that affect how the SOM classifies vertices by using different training data sets features and varying the SOM's size and training parameters. Curvature metric combinations not previously used when identifying surface types are used as features. Two different training data sets are used, the object being classified and a set of exemplar paraboloids used to represent different conic sections. We apply this classification technique to the paraboloid training data set and a complex free-form surface. The distribution of neuron activity and proportion of neurons fired during the classification process, mean quantisation error and visual inspection are used to identify the affect of each of the experimental parameters on the classifier output.

In Chapter 4 we apply the knowledge from the previous chapter to use the SOM clas-

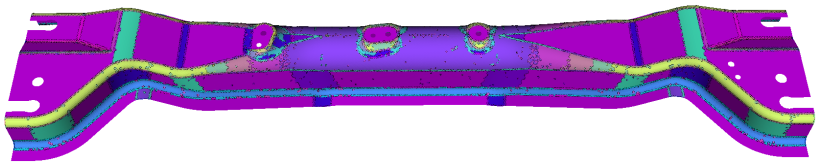
sifier output as the basis of a free-form surface segmentation algorithm. We conduct parameter sensitivity experiments using the same methods as the previous chapter using random free-form surfaces, paraboloids from the previous chapter and the model being segmented as training data. We find that when segmenting 3D models the model itself being segmented is the best source of training data as it delivers the most consistent results. Figure 1-1 illustrates examples of free-form surfaces segmented using our approach using a colour map to identify regions that share the same or similar cluster on the SOM. Our models are rendered with shading to make their 3D structure clear to the reader and this rendering will introduce differences in brightness on the surface of the model.

In Chapter 5 we investigate the indexing and retrieval of free-form surfaces by taking a ”bag-of-words” (BoW) inspired approach, using the classifier output to describe regions and whole surfaces as histograms of vertex-type distributions. The simple BoW approach discards spatial relationships between regions and we will rely on the effectiveness of our classifier to match similar regions, much like a human will recognize key words in a document while scanning it quickly for relevance and implemented in document classification techniques [IEE, 2009]. To make our technique reasonably tolerant to scale variance we will take multiple samples using a sphere as a selection widget with multiple radii centered at each vertex of the mesh representation of the surface when calculating our histogram representation. As with the previous chapter we evaluate the affect SOM and its training parameters have on the indexing and retrieval problem with the addition of two distance metrics, least mean squared difference (LMS) and the Earth-Movers Distance (EMD). Region matching experiments are performed on noisy 2.5D range-data, a 3D model reconstructed from multiple views and CAD models and we then investigate the performance of our descriptor to describe object classes using a CAD model benchmark dataset. We then applied our technique to identify object classes using a CAD benchmark dataset.

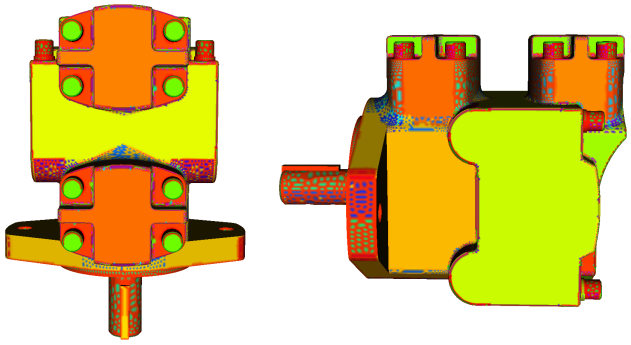
The conclusion consists of a summary of our findings with an examination of the strengths and weaknesses of our approach and we will present ideas for future research.



(a) Spigot[New Dimension Systems, 2011]



(b) Cross-member



(c) Vane motor[New Dimension Systems, 2011]

Figure 1-1: Segmented free-form surfaces and CAD models using our SOM based technique

Chapter 2

Background and Related Work

The indexing and retrieval of free-form surfaces requires breaking down surfaces into descriptor sets that can be used to build an index. Using clustering techniques the vocabulary required to encode these descriptors can be reduced making it simpler to break down surfaces into their components to describe their parts, build an index or segment surfaces.

One of the simplest representations of free-form 3D surfaces that can be easily derived from other formats such as range images and NURBS is a polygonal mesh made up of triangles. We will be using this representation throughout this dissertation.

Curvature is a basic property that describes the rate of change in direction of a line and can be calculated at any point on a continuous 3D surface, a pixel in a range image and vertex from a triangulated mesh. The ubiquity of this measure lends itself to being used as a feature to describe any surface regardless of its representation. An advantage of using a surface based feature over a volume based descriptor is that it can be applied to both solid 3D models and free-form surfaces such as sheet metal pressed into car body panels.

This chapter covers the relevant background material for the thesis, we detail the workings of the Self-Organizing Map, give an overview of the curvature metrics that we will use as features and an overview of the current state of the art in 2.5D and 3D free-form surface segmentation and indexing and retrieval methods. The many different methods for segmenting free-form surfaces are presented. Finally techniques for indexing and the retrieval of 3D objects are presented.

2.1 Self-Organizing Maps

Kohonen's biologically inspired Self-Organizing Map (SOM) is an unsupervised, self-organizing artificial neural network that changes its neurons during training to more

closely match the input data used as stimuli. After training the neurons are topologically ordered so adjacent neurons share similar properties. The SOM can deal with noisy input data [Haritopoulos et al., 2002, Sucher, 1995] and dynamic signals such as sampled speech and processing can be performed in parallel and can deal with degraded input signals (ie. incomplete data sets) gracefully. This powerful technique can find non-linear and non-parametric multivariate relationships in data sets that can contain significant amounts of noise. The SOM is a powerful tool for visualising these relationships in order to develop a better understanding of how input data sets are clustered.

We chose an unsupervised learning technique over supervised and reinforcement learning because we wanted to automate as much of the learning process as possible with no intervention from a human critic and with no preconceived notions of how regions should be categorised. Unsupervised learning enables us to efficiently process very large training data sets without being bound by learning time constraints or biases introduced by human interaction. The benefit of using a SOM as our classifier compared to other unsupervised learning techniques is the ability to effectively visualise relationships between clusters making interpretation of the classification process easier to understand and improve.

The SOM has been used to solve a wide variety of task such as compression [Abidi et al., 1994, Barbalho et al., 2001], machine learning and vision [Beard and Rattan, 1989], texture analysis [Parada and del Solar, 2001], speech recognition, hand written character recognition [Aksela et al., 2003] and segmentation [Yao et al., 2000]. This list is a tiny sample of what can be achieved with this powerful technique. Given the SOM's ability to deal with complex multivariate relationships, noisy data and the powerful visualisation techniques that can be applied to problems it is no surprise that there are over 7000 citations of SOM related works in the SOM bibliography alone [Pöllä et al., 2008].

Research has shown that a SOM combined with surface based metrics can be used to segment 2.5D range data [Koh et al., 1995, Bhandarker et al., 1997] and we can build on that success to use the SOM to describe, segment, index and identify object classes of

free-form surfaces.

2.1.1 Artificial Neural Networks

Artificial neural networks (ANNs) were originally created to emulate how the brain operates. The brain is able to process multiple stimuli in parallel using prior experience to make decisions. This experience is recorded using the connectivity strength and response within the neural structures of the brain. In ANNs the neurons are approximated using mathematical functions and their connectivity is defined using layers and inter-neural connections.

As a deeper understanding of the biochemical and electrical systems of the brain developed the computational requirements to emulate these systems became too great at that time [Minsky and Papert, 1969]. The focus of research into ANNs then moved away from modelling brain function directly and instead applied the these models to problems where traditional existing methods have failed or were computationally expensive. In the context of ANNs a Neuron is represented by a mathematical function that gives a response based upon the stimuli.

Kohonen [2001] summarised their similarities with the brain as follows:

- *Analogue representation and processing of information*
- *Ability to average conditionally over sets of data*
- *Fault tolerance as well as graceful degradation and recovery from errors*
- *Adapting to changing environment and emergence of "intelligent" information processing functions by self-organization, in response to data.*

These attributes allow the ANN to process data that is noisy, asynchronous in nature, that contains non-parametric distributions and relationships. ANNs are also able to take

into account the collective properties of the data set, so eigenvalue like properties can be represented, and during the learning process the ANN will adapt itself to the input data and create detectors for commonly occurring features in the data set [Kohonen, 2001]. All ANN learning processes follow these basic steps [Haykn, 1994, pp.45];

1. Stimulation
2. Changes in the ANN in response to the stimulation
3. A different response to the original stimulation reflecting the changes made to the ANN in the previous step

Numerous learning algorithms for ANNs have been developed over time and can be categorised into three different learning paradigms; supervised, reinforcement and unsupervised learning.

Supervised ANNs requires a domain expert in the training process to guide the ANN to the correct solution given the current state of the system. When training an ANN using this technique the information used is coupled with the required output of the system. A closed feedback-loop system is used to change the ANN so it more closely matches the desired output. Training can take place before the ANN is used or it can evolve over time in a live environment. These systems perform well for problems where the training data can cover all of the required responses of the system, but is not able to infer previously knowledge unknown to the expert used to train the system [Haykn, 1994, pp. 59].

Reinforcement learning ANNs dictate how decisions are made by framing the problem in terms of situations, possible actions and a reward metric that a software agent has to take into account when making decisions. The two attributes of reinforcement learning that differentiate this learning technique from others is the reward for making decisions has an immediate and delayed effect, and the search process for choosing the best decision to maximise the reward is by trial and error [Sutton and Barto, 1998]. This goal-oriented form of learning has to find a balance between using existing knowledge and exploring

the outcome of unexplored decisions. Reinforcement learning is well suited to problem domains such as game playing and robotics and control systems where there is a long-term goal to be achieved which requires both short and long term rewards to be taken into account [Kaelbling et al., 1996].

Unsupervised learning, also known as Self-Organizing, ANNs only require input data to train the network, and is trained without any external assistance or guidance. Algorithms are used to determine how well the neurons represent the training data and how the neurons should be modified to more closely match the input data. After the ANN has been stimulated with enough data the neurons are able to discern between different clusters in the data set.

Competitive learning techniques use the concept of winner takes all when stimulating the ANN. A unit of information, the stimulus, is passed into the ANN and the neurons compete with each other to see which one best represents the stimulus, only the neuron that best represents the stimulus fires.

Some of the benefits of using unsupervised competitive learning techniques compared to supervised and reinforcement learning are:

- Neurons become specific feature detectors
- Non-linear and non-parametric relationships are discovered
- The learning and classification processes require no user intervention
- There is no need for external experts to guide the learning process

2.1.2 The Self-Organizing Map

The Self-Organizing Map (SOM) was conceived by Teuvo Kohonen Kohonen [2001] to reduce complex multivariate non-parametric relationships into a simpler 2D or 1D repre-

sentation that can easily visualised. The SOM is a single layered artificial neural network that consists of a single layer of interconnected neurons arranged in a fixed topology that employs competitive learning techniques to perform unsupervised learning. The core benefit of using the SOM to cluster multivariate data is that complex relationships are described using a simple representation that is topologically ordered.

The SOM training data does not require any manual tagging as the process is totally unsupervised. The learning algorithm alters the neurons of the SOM to more closely resemble the training data and is controlled using decay functions that determine the range and strength the stimulus has on the SOM's neurons. For our investigations we implemented the algorithms in Java required to describe, train and classify data using a SOM.

Following the notation in Kohonen [2001] we will now formally describe the SOM and its learning process.

A neuron in a SOM consists of a vector of values called the model vector that is used to represent a class of stimuli. The model vector m_i, where i is the index of that neuron, and n is the number of values in the vector, is described as:

$$m_i = [\mu_1, \mu_2, \mu_3 \dots \mu_n]^T \in \mathbb{R}^n \tag{2.1}$$

The corresponding input vector that is used to stimulate the SOM is:

$$x = [\xi_1, \xi_2, \xi_3 \dots \xi_n]^T \in \mathbb{R}^n \tag{2.2}$$

In order to determine which neuron best describes the input vector a distance function is used, the Euclidean distance is the most commonly used metric for this task. To find the closest matching neuron, also known as the best matching unit (BMU), given an input

vector we find the neuron c that has the least distance between its model vector and the input vector:

$$c = \overset{\text{argmin}}{i} \{\|x - m_i\|\} \tag{2.3}$$

We exclude any infinite or undefined values when calculating the distance between feature vectors.

Using these functions we can now describe the operation of the incremental SOM. The incremental SOM update is an iterative process where the model vectors are modified to more closely resemble the input vectors. the weights of the neuron c and its neighbouring neurons are updated.

The update function causes topologically local neurons of the SOM to share similar weights and over time the SOM will exhibit a global ordering [Kohonen, 2001]. This topological ordering was inspired by the ordering found in the brain where regions of the brain can be mapped to different process and function types. The update function for a given time period t and neighbourhood function h_{ci}, where i is the index of the current neuron being updated and c is the winning neuron described in the previous equation is defined as:

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)] \tag{2.4}$$

The neighbourhood function h_{ci} is a kernel operator that determines to what extent other neurons in the SOM are affected by the update algorithm. For the neurons to converge on a stable set of values the limit: $\lim_{t\to\infty} h_{ci} = 0$ must hold. The Gaussian distribution combined with a decay function is commonly used as a neighbourhood function in SOMs:

$$h_{ci} = \alpha(t) \exp(-\frac{\|r_c - r_i\|^2}{2\sigma^2(t)}) \tag{2.5}$$

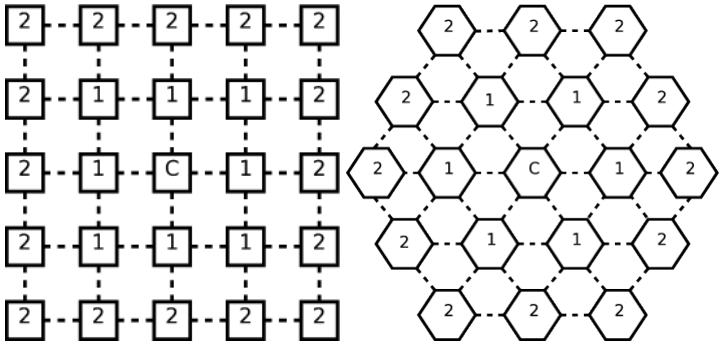
Both $\sigma(t)$ and $\alpha(t)$ are decay functions that decrease over time t. The σ function determines the width of the kernel and the α function dictates how strong an affect the input vector has on the neuron being updated. The two training phases, coarse and fine, use the same update and neighbourhood functions. The difference is that the coarse training update function affects a large portion of the SOM and the fine training phase focusses its update function on smaller regions around the BMU. The coarse training process is intended to enforce global ordering amongst the neurons and the fine training process is used to tune the SOM after the initial training. Our implementation of the SOM will use an initial update function size that covers the whole SOM for the coarse training phase and fine training phase will initially affect 20 percent of the SOM. Both of these areas decay over time using Equation 2.5.

Training the SOM with only the coarse parameters may result in an underfitted SOM that will converge on a solution quickly. The process of passing every element of our training data set once into the SOM for training is called an epoch. If we only use the fine training process it will take a larger number of epochs to converge onto a solution and result in an overfitted SOM because the parameters are focussed on small regions and will update them frequently due the number of epochs required. The example in Figure 2-1 illustrates how quickly the coarse training parameters converge on a solution compared to the fine training epochs. Similar colours on this figure indicate regions of similar curvature. We map these colours to neurons on the SOM using the colour lookup table in Figure 3-3 on Page 58. The image representing the result of a single fine training epoch shows shaded regions that are artefacts of the rendering we are using to highlight the 3D structure of the model, there are no distinct regions on this model.

For the neighbourhood function to operate we need to understand how the neighbourhood radius values map to the neurons. The neurons are only connected to their immediate neighbours and the connections are arranged in a lattice, the most common types

(a) 1 coarse and 0 fine training epoch (b) 0 coarse and 1 fine training epoch

Figure 2-1: Comparison of vertex classification results after training a SOM with a single coarse or fine training epoch.



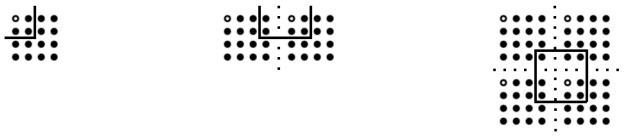
(a) Rectangle (b) Hexagon

Figure 2-2: SOM Neural Topologies

being hexagonal and rectangular. The hexagonal neighbourhood is preferred as there are a larger number of links between the neighbouring neurons compared to the rectangular arrangement and they are all equal in distance. These differences are illustrated in Figure 2-2.

The original shape of the SOM topology was a projection onto a sheet and this affected the learning process as neurons that were on the border of the sheet had less neighbouring neurons, affecting the effectiveness of the SOM update function. This problem is known as the "Border Effect" and to alleviate this problem Kohonen [2001] altered the weighting of neurons at the boundary and others used different shapes, such as the sphere and torus [Wu and Takatsuka, 2006, Oy et al., 2000] and hyperbolic space [Ritter, 1999].

Figure 2-3 illustrates how the neighbouring neurons are determined for the sheet, cylin-



(a) Sheet (b) Cylinder (c) Torus

Figure 2-3: Nearest neighbouring neurons calculation with a rectangular lattice

der and torus. The centre neuron is hollow, the solid line encloses the neighbouring neurons and the dashed line represents where shape causes the lattice to connect on the edges, i.e. the top of the lattice connects to the bottom of that same lattice in Figure 2-3(c). The toroidal shape is preferable because for each given neuron there are the same number of immediate neighbouring neurons, avoiding any border effects.

Before the SOM can be used the neurons need to be initialised, the process can be linear, random or analytical. Random initialisation assigns random vales to each neuron's weights and the linear initialisation process assigns values in constant increments. Analytical initialisation processes use techniques such as principal component analysis (PCA) to determine initial neuron weights so they coincide with clusters in the data set [Attik et al., 2005]. We chose a linear initialization technique to ensure a consistent SOM state before training for all of our experiments.

Once a SOM has been trained the quality of the representation can be numerically evaluated by measuring the quantisation error (Q) for the training data set. The quantisation error of a single feature vector is the Euclidean distance between that vector and its best matching neuron (BMU). Throughout the thesis we will evalute the mean Q for an entire feature set. A low Q usually indicates the SOM is a good representation of the training data, but if the quantisation error is very small then the SOM has been over-fitted to the data set. SOMs have no way to measure noise in the incoming data so it will attempt to represent valid and noisy data. This problem is illustrated in the results from training a SOM on noisy range data in Section 5.1.9 on page 189.

Where x is the feature vector being evaluated, m the best matching neuron for x and n is number of values that make up the neuron and feature vector the calculation of Q is described in Equation 2.6.

$$Q = \sqrt{\sum_{i=0}^{n}(x_i - m_i)^2} \tag{2.6}$$

2.1.3 Visualising the Self-Organizing Map

One of the most useful properties of the SOM is its ability to learn and project complex multi-dimensional relationships onto a 2D plane. According to Vesanto [1999] the three relationship types that need to be visualised in order to understand how the clusters and input data relate to each other are: (1) The shape and general cluster structure of the SOM, (2) The characteristics of the neurons and correlation between neurons components, and (3) The evaluation of data samples when classified by the SOM.

To demonstrate SOM visualisation techniques we will use sample code and output from the SOM Toolkit [Vesanto et al., 1999b]. Our dataset is a set of points (x, y, z) in $\mathbb{R}^3$ space created by sampling from Gaussian distributions centered around three points, $(0, 0, 0)$, $(3, 3, 3)$ and $(9, 0, 0)$. The cluster memberships in Figure 2-4 are denoted in red, green and blue with the SOM neuron vectors marked with the + sign. This figure demonstrates how difficult it is to determine which neuron each of the data points belongs to, distance in feature space between neurons and how this membership is distributed when viewing a plot of multidimensional data. Our SOM implementation outputs the SOM structure and hit statistics in a format readable by the SOM tool kit so we could re-use the visualization code already implemented in this software suite.

To visualise the SOM's structure multidimensional scaling (MDS) techniques such as principal components analysis (PCA) (Figure 2-6) and Sammon's Mapping are used to project the neural relationships using points and lines in $\mathbb{R}^2$ or $\mathbb{R}^3$ space using a colour

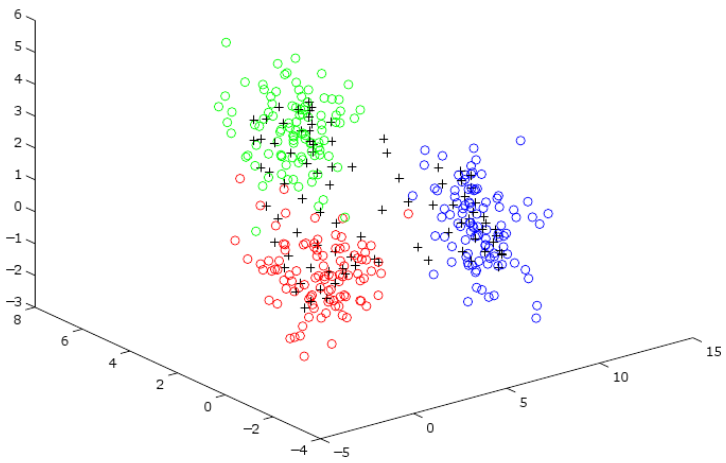


Figure 2-4: Example dataset of points in $\mathbb{R}^3$ space [Oy et al., 2000]

map to relate the point to a neuron. If neighbouring neurons are highly dissimilar then the projection into a lower dimension space will be complicated making the relations difficult to visualise.

The U-matrix [Kohonen, 2001], distance matrix and similarity colouring techniques are a more effective way to show the feature distance between neighbouring neurons. The U-matrix uses shades of grey in between each neuron to indicate distance. The distance matrix uses the size of the hexagon shaped icon that represents the neurons to indicate average distance between each neuron and similarity colouring uses colours to indicate the feature distance between each neuron. The distance matrix and similarity colouring are two other techniques that can be used to visualise the SOM's structure and shape. The distance matrix uses the size of a hexagon indicating neuron position or shades of grey to show the distance between each neuron and similarity colouring uses colour to indicate distance or similarity. These three techniques using our sample data set are illustrated in Figure 2-5. All three visualisation techniques show three distinct clusters, the lower left, upper left and right hand regions separated by a dark line, which is what we would expect given how the distribution of data points used to train the SOM were created.

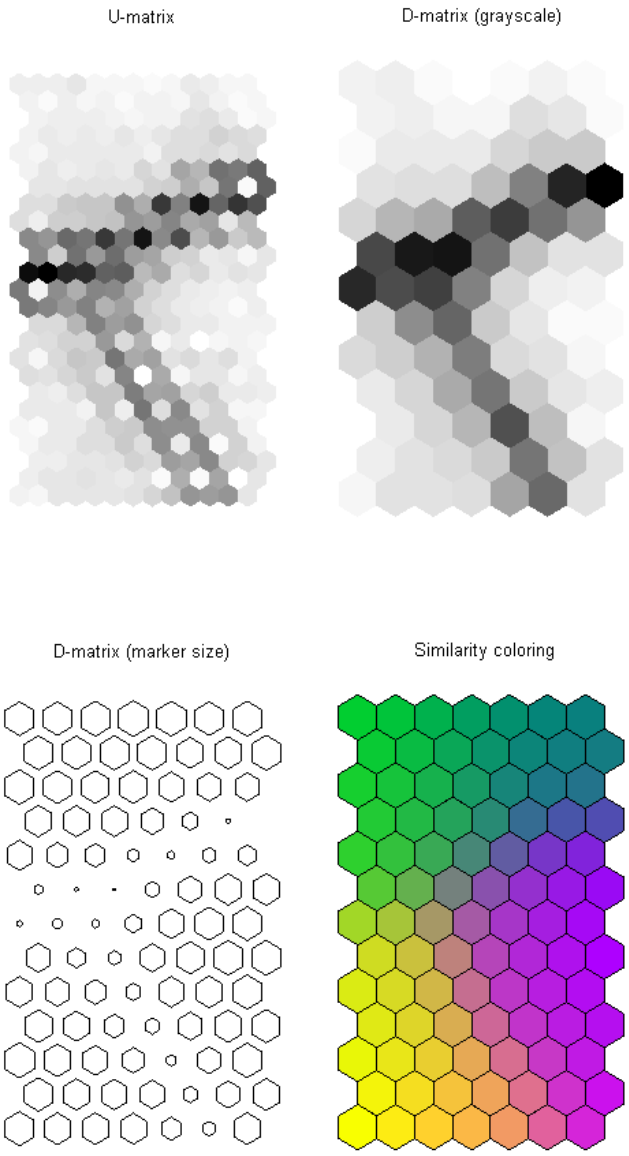


Figure 2-5: Inter-neural distance visualisation techniques taken from examples in Oy et al. [2000]

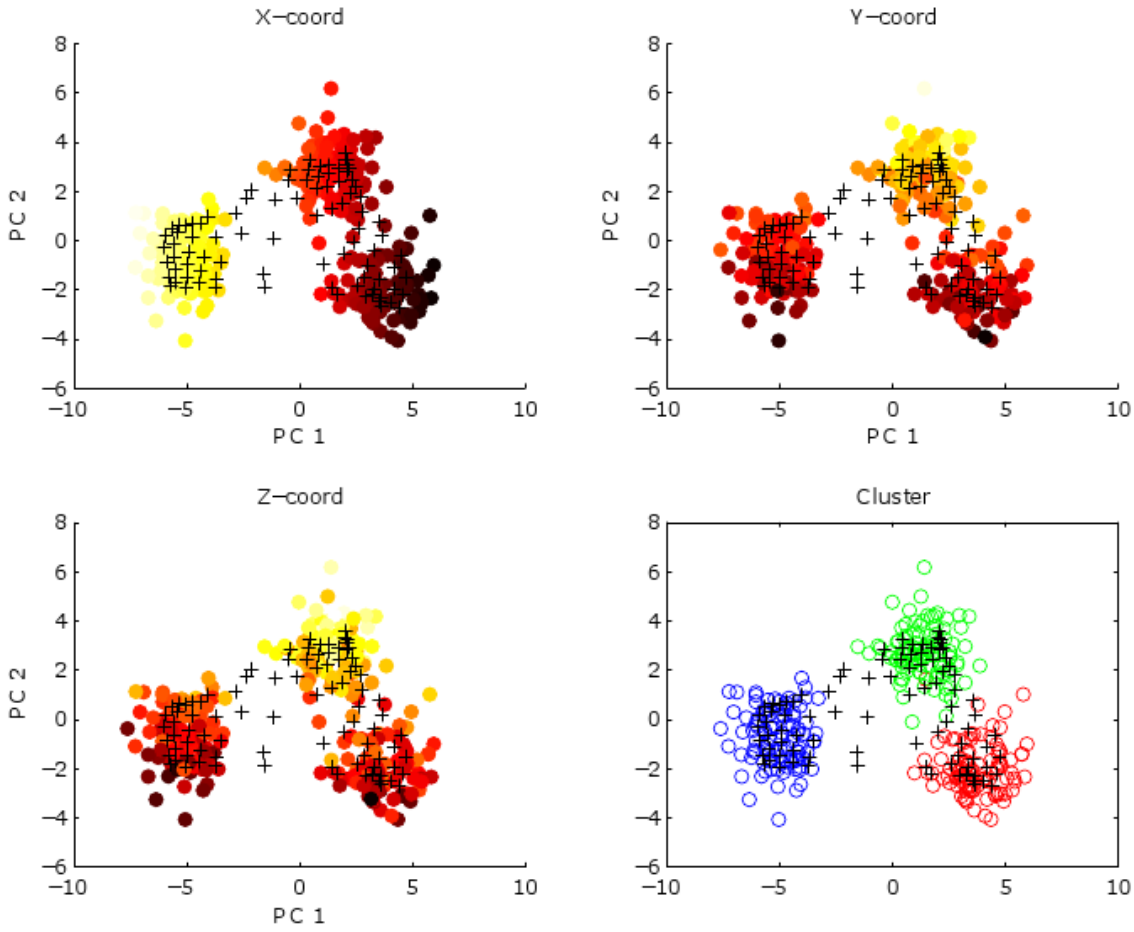


Figure 2-6: PCA Projections of relationships between the data points in Figure 2-4 [Oy et al., 2000]

When using a traditional projection technique such as principal components analysis (PCA) to visualise clusters the position or colour of a data point is used to link that data point across the different plots. In Figure 2-6 the points are related to each other using position and their value is denoted by colour. It requires more effort to understand that there are three distinct clusters in the dataset when using three plots using PCA compared to the single plots illustrated in Figure 2-5.

To understand how the neurons differ from each other on a per-feature basis the individual components x, y and z are plotted on their own component planes using the U-matrix

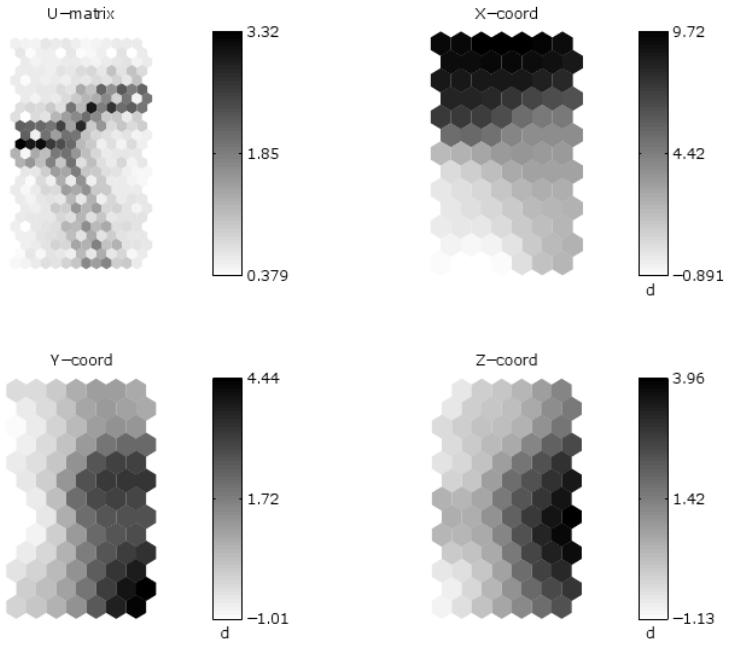


Figure 2-7: The U-matrix and component planes of x, y and z [Oy et al., 2000]

to show the distances between these individual features. The component planes and U-matrix in Figure 2-7 show that the cluster in the upper region is dominated by large x values and low y and z values, which would make sense since one of the centre points used to generate the data set is at $(9, 0, 0)$. The cluster of neurons to the lower left in the U-matrix has small values for all three components, so this cluster would represent the data centered around $(0, 0, 0)$. The final cluster on the right hand side of the U-matrix shows high y values with average x values and high z values, these component attributes would describe the values centered at $(3, 3, 3)$. These examples demonstrate the effectiveness of the U-matrix for visualizing inter-neural distances and SOM responses to stimuli.

When using the SOM to classify input data we can project the sum of the neuron responses onto the U-matrix of the SOM. This technique allows us to quickly identify how input data is distributed across the clusters defined by the SOM. The example in Figure 2-8 uses red hexagons to indicate which neurons have responded to the input data and what proportion of the input data that neuron most closely matched. The U-matrix on the left shows the response of the SOM given 50 data points that are clustered around

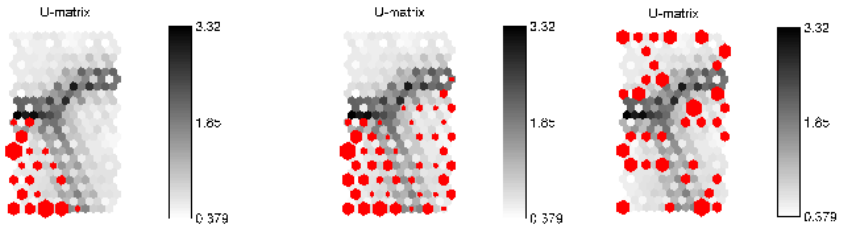


Figure 2-8: Neuron response plots for data set samples [Oy et al., 2000]

the point $(0, 0, 0)$ and shows that the cluster of neurons that describe these points is the lower left corner of the U-matrix bounded by the dark regions that denote large inter-neuron distances. The centre U-matrix projection shows sample data taken from around the points $(0, 0, 0)$ and $(3, 3, 3)$ that identifies the cluster of neurons in the lower right hand region as responsive to data sampled from around the $(3, 3, 3)$ point. The neural response illustrated on the right hand side is the result of taking a random sample from the set of of all data points.

We have demonstrated that the SOM is able to find and visualise the three clusters generated from three distinct data sets. The visualisation techniques demonstrated in this Section show that the ordering of the SOM matches the ordering of the synthetic data sets using the U-matrix and component diagrams. The SOM has been shown to be a versatile learning and visualisation technique that can be applied to wide variety of complicated problems.

2.2 Curvature

Curvature is an intuitive property that we use to describe objects around us every day, for example we would describe the surface of a road as flat, a round ball and a sharp edge. For curves it is the rate of change of the direction of the curve and is defined for a point p on a curve s using using the unit tangent vector $\mathbf{T}$ as $k = |\frac{d\mathbf{T}}{ds}|$ [Stewart, 1999].

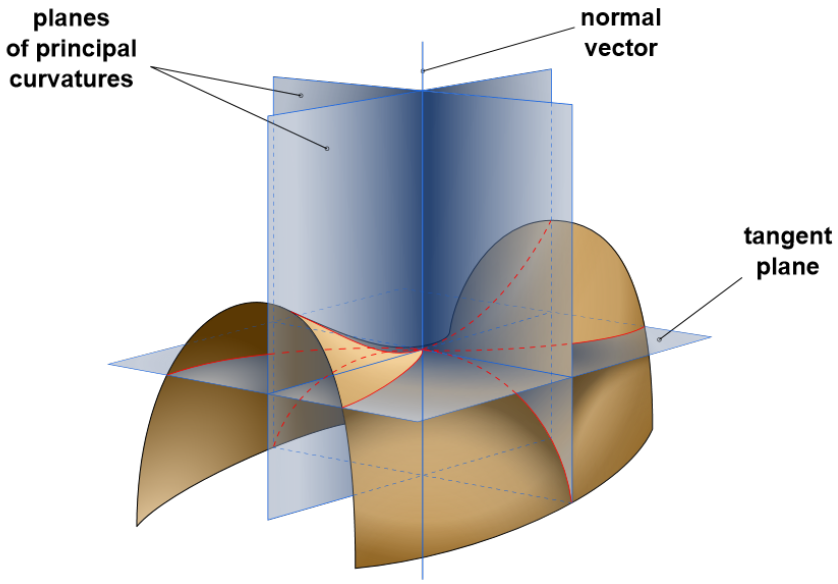


Figure 2-9: Principal curvatures and direction of a saddle. Image created by Gaba [2006].

	$k_1 < 0$	$k_1 = 0$	$k_1 > 0$
$k_2 < 0$	peak	ridge	saddle
$k_2 = 0$	ridge	flat	valley
$k_2 > 0$	saddle	valley	pit

Table 2.1: Surface types from k_1 and k_2 [Besl and Jain, 1988]

The curvature of a free-form surface, known as normal curvature k, is defined for a point p and angle θ on a surface s. It measures the derivation from the tangent plane at p (also known as the osculating plane) by the line formed from the intersection of the surface s and a plane defined as the normal of p at the angle θ [Meyer et al., 2002, Weisstein, 2012a].

Principal curvatures of a surface at p are the maximum k_1 and minimum k_2 normal curvatures for all angles around the point. The planes and lines used to define the k_1 and k_2 of a saddle are shown in Figure 2-9 with the lines in red illustrating the maximum and minimum normal curvatures. Besl and Jain [1988] were able to derive surface types from principle curvature signs. This mapping is shown in Table 2.1.

	$K < 0$	$K = 0$	$K > 0$
$H < 0$	peak	ridge	saddle ridge
$H = 0$	(none)	flat	minimal
$H > 0$	pit	valley	saddle valley

Table 2.2: Surface types from H and K [Besl and Jain, 1988]

Mean curvature is the mean of all of the normal curvatures at a point p and can be calculated by finding the mean of k_1 and k_2. It is an extrinsic curvature metric measured in the context of the space that the surface has been embedded in [Koenderink, 1990].

$$H = \frac{1}{2}(k_1 + k_2) \tag{2.7}$$

Gaussian curvature is an intrinsic curvature metric, measurable at any point on the surface independently of the space that surface is embedded in [Koenderink, 1990] and describes the surface at p as elliptic, hyperbolic, parabolic or planar [Gray, 1998]. It can be calculated using the product of the principal curvatures k_1 and k_2.

$$K = k_1 k_2 \tag{2.8}$$

In the same way k_1 and k_2 were combined to describe surface types, Besl and Jain [1988] used the signs of Gaussian (K) and mean curvature (H) to describe the surface type at a point p as shown in Table 2.2.

2.2.1 Measuring Curvature

The most common representations of surfaces for visualization and processing are triangulated meshes, encoded as edges and vertices, as they are simple to describe and

manipulate. When triangulated meshes are created from scanned data the output can be noisy due to artifacts introduced in the data acquisition process. Transforming surfaces described by mathematical functions into meshes can be problematic, e.g. detail loss from low sampling rates and non-preservation of sharp edges. These factors can make calculating curvature metrics a difficult proposition for triangulated surfaces compared to mathematically defined free-form surfaces.

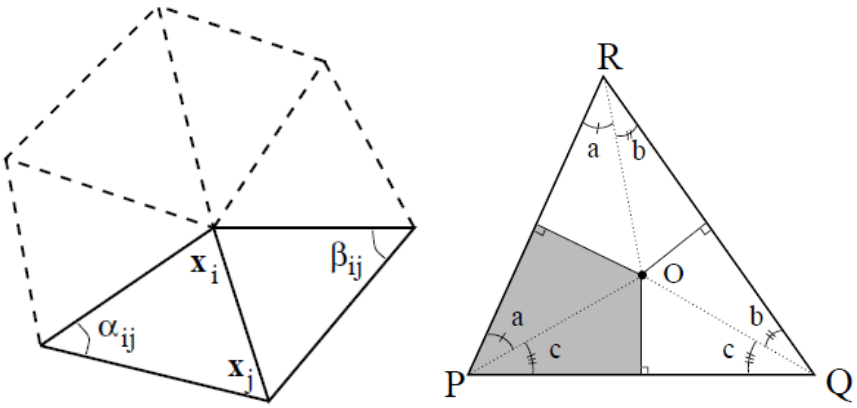
Meyer et al. [2002] approached the problem of calculating curvature metrics for triangulated surfaces using a finite-element/finite-volume approach using a local 1-ring neighbourhood of connected vertices around the point p being evaluated. It was shown to produce results similar to continuous cases for reasonably smooth surfaces with a regular triangulated mesh. This approach allows us to calculate the curvature of vertices that appear as sharp edges as the curvature is calculated as a 1-ring neighbourhood and not as a single point. We implemented each of the algorithms in Java using the algorithms described in this Section.

Using the notation described earlier in this section the mean curvature for a continuous surface is defined as the sum of normal curvatures for each angle θ at p:

$$H = \frac{1}{2\pi} \int_0^{2\pi} k(\theta) d\theta \tag{2.9}$$

When this equation is equal to zero it becomes the Euler-Lagrange surface area minimization function which is directly related to mean curvature flow [Meyer et al., 2002]. This relationship is used to calculate the Mean curvature H using properties of an infinitesimally small area $\mathcal{A}$ around point p with the gradient $\bigtriangledown\mathcal{A}$ at p and is expressed as:

$$2H = \lim_{diameter(\mathcal{A}) \to 0} \frac{\bigtriangledown\mathcal{A}}{\mathcal{A}} \tag{2.10}$$



(a) 1-ring neighbourhood and opposite edge angles.

(b) Voronoi region of a triangle.

Figure 2-10: Region definitions used to calculate the area around a point reproduced from Meyer et al. [2002] ©2003 Springer.

To calculate H for a triangulated surface a mean position within each triangle that has vertices connecting to p is used as the finite element. The finite volume is represented by the area $\mathcal{A}_{\text{Mixed}}$ which is the sum of the surface areas for each triangle that makes up the 1-ring neighbourhood N_1 of connected vertices around p. The area of each triangle in the 1-ring is calculated for non-obtuse triangles using voronoi cells (Figure 2-10(b)) or half or a quarter of the area of an obtuse triangle depending on if the angle at p is obtuse or not.

Meyer et al. [2002] derived the function $K(x_i)$ to calculate the mean curvature H at the point x_i (p in our notation). Figure 2-10(a) illustrates the edges, vertices and angles used in the function. The two angles α_{ij} and β_{ij} are on opposite sides of a line $x_i\vec{x}_j$ where x_i is the vertex being evaluated for mean curvature normal and j exists in N_1. The function is defined as:

$$K(x_i) = \frac{1}{2\mathcal{A}_{\text{Mixed}}} \sum j \in N_1(i)(\cot\alpha_{ij} + \cot\beta_{ij})(x_i - x_j) \tag{2.11}$$

Gaussian curvature for p can be calculated as a limit using the spherical image $\mathcal{A}^{\mathcal{G}}$, a projection of the surface around p onto a unit sphere, and area $\mathcal{A}$:

$$K = \lim_{diam(\mathcal{A}) \to 0} \frac{\mathcal{A}^{\mathcal{G}}}{\mathcal{A}} \tag{2.12}$$

The discrete Gaussian function K_G finds the difference between 2π and the sum of the angles for each face around the 1-ring neighbourhood around vertex x_i and is defined as:

$$K_G(x_i) = (2\pi - \sum_{j=1}^{\#f} \theta j)/\mathcal{A}_{\text{Mixed}} \tag{2.13}$$

Using the discrete functions for calculating H and K, Meyer et al. [2002] used substitution to calculate k_1 and k_2. For all of our curvature metrics we will use an ϵ value of $1.0e^{-10}$ to round values to 0 unless otherwise noted.

[Koenderink, 1990] used k_1 and k_2 to derive two curvature metrics, curvedness C and shape index S:

$$S = -\frac{2}{\pi}\text{arctan}\frac{k_1 + k_2}{k_1 - k_2} \tag{2.14}$$

The shape index S describes how the surface is curved at a given point. It is a continuous measure defined for all surface types except for planes and is invariant to changes in scale. Our implementation maps the value to the range $[-1, +1]$. Convex surfaces have a positive sign and concave are negative. Regions with $S = \pm 1$ are convex and concave umbilics (locally spherical) and when $S = 0$ they are minimal points or symmetrical saddles [Koenderink, 1990]. Dorai and Jain [1995] mapped the function to the interval $[0, 1]$. The ranges and surface categories for this mapping is shown in Figure 2-11. We applied an epsilon value of 1.0^{-4} to k_1 and k_2 when calculating S.

Curvedness C works in concert with the shape index when describing a surface. It describes the magnitude of the shape index S and is defined in Algorithm 2.15. Zero indi-

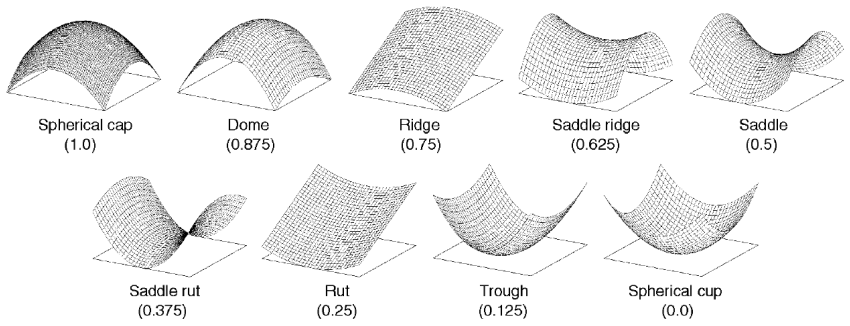


Figure 2-11: Shape index surface categories taken from Dorai and Jain [1995] ©IEEE

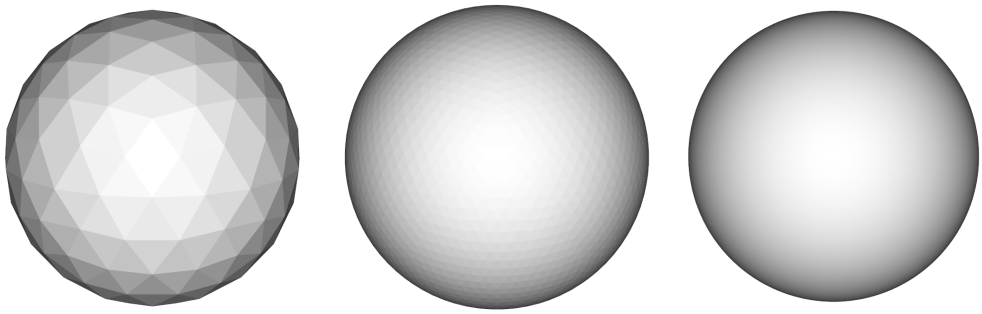
cates a planar region, low values a subtle bend of the surface and high values denote high curvature such as the point or edge of a knife. Combining the two measures is akin to a polar mapping using S as the angle and C as the magnitude. The benefit of using C and S over other measures such as H, K, k_1 and k_2 is that the curvature type is decoupled from the magnitude of the curvature [Dorai and Jain, 1995].

$$C = \sqrt{(k_1^2 + k_2^2)/2} \tag{2.15}$$

Dorai and Jain [1995] applied S and C to the problem of identifying free-form surfaces from range data by finding maximal patches that share the same S and projecting the mean normal of these patches onto a sphere. Each of the points representing a connected patch was annotated with properties such as C and patch area. This representation was used to cluster range images of like objects in Dorai and Jain [1997b].

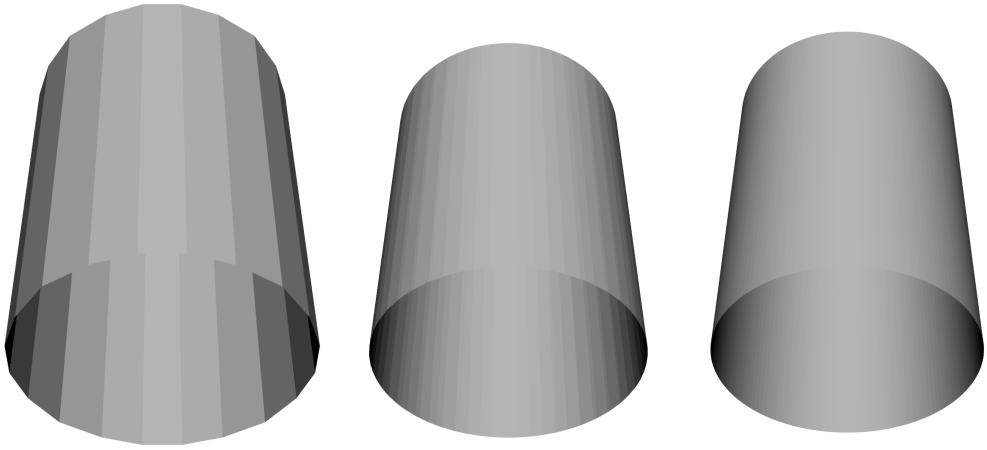
2.2.2 Meshing and noise

To measure the impact that meshing parameters and noise have on curvature measurement we used an unit sphere (Figure 2-12) and cylinder (Figure 2-13) , adjusted the meshing parameters and examined the curvature distributions to see how they changed. The



(a) 162 vertices (b) 2562 vertices (c) 40962 vertices

Figure 2-12: Sphere meshed at varying levels of detail



(a) 240 vertices (b) 2160 vertices (c) 28116 vertices

Figure 2-13: Cylinder meshed at varying levels of detail

Vertex count	k_1	k_2	K	H	S	C
162	0.71	0.58	1.02	1.02	0.77	0.65
2562	0.48	0.44	1.00	1.00	0.72	0.46
40962	0.07	0.07	1.00	1.00	0.53	0.07

Table 2.3: Summary of curvature metrics for spheres meshed at varying levels of detail.

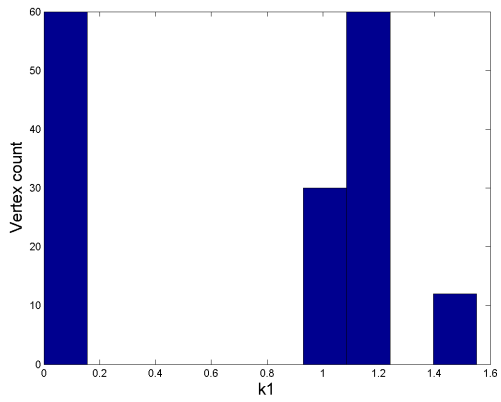
Vertex count	k_1	k_2	K	H	S	C
240	0.73	0.73	0.73	0.73	0.73	0.52
2160	1.31	0.00	137.30	0.75	0.50	0.93
28116	1.18	0.00	1932.67	0.61	0.50	0.83

Table 2.4: Summary of curvature metrics for cylinders meshed at varying levels of detail.

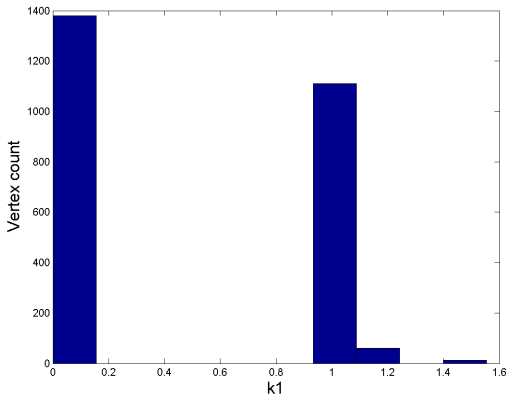
effect noise had on the discrete curvature measurements were evaluated using the unit sphere and adding noise to the (x, y, z) values of each vertex.

To understand how the meshing parameters affected the curvature metrics calculated using the techniques described by Meyer et al. [2002] we calculated the mean values for k_1, k_2, K, H, S and C for both surfaces (Tables 2.3 and 2.4.) The principle curvatures k_1 and k_2 of a sphere should always be equal as the curvature in all directions is the same. As the number of vertices used to represent the surface increases they converge on a similar value. Figure 2-14 shows the convergence of the mean toward 0 with less outliers in the finest level of detail compared to the two coarser representations. This convergence is also evident for both K and H as they stabilise at 1.0 which is the expected values for an unit sphere.

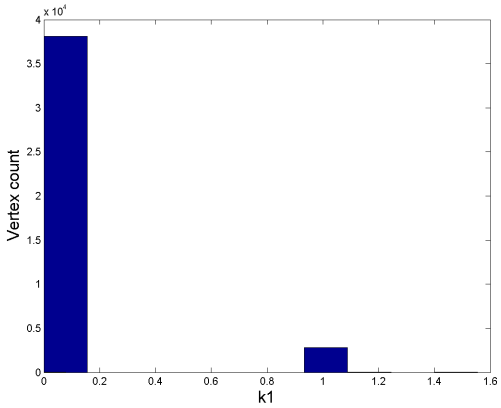
The shape index S appears to be at its most accurate value when the mesh is at its coarsest and then starts to diverge as the k_1 and k_2 values get smaller. This suggests that the calculation for S is less accurate as the values approach the extremes of the range $[-1, +1]$. Curvedness C measures the magnitude of the curvature component when calculating S and is very similar to k_1 and k_2 at all levels of detail.



(a) 162 vertices

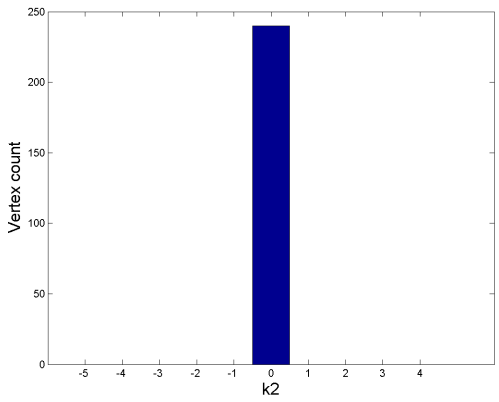


(b) 2562 vertices

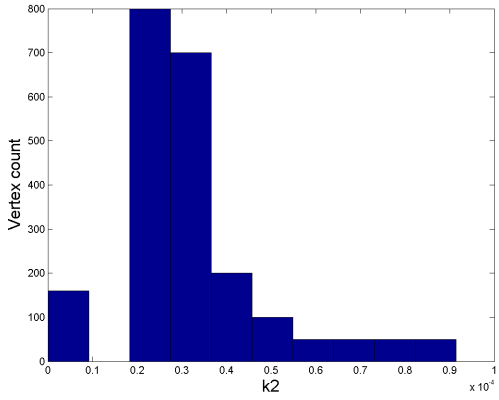


(c) 40962 vertices

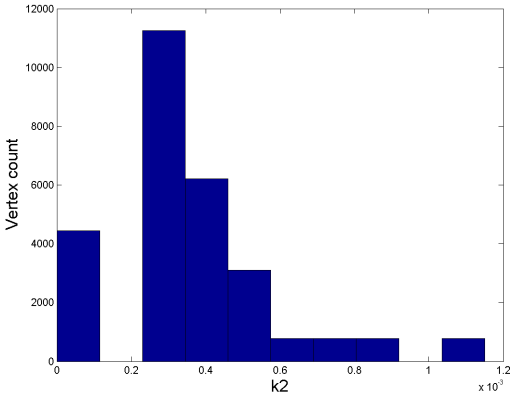
Figure 2-14: k_1 distributions for the Sphere in Figure 2-12



(a) 240 vertices



(b) 2160 vertices



(c) 28116 vertices

Figure 2-15: k_2 distributions for the Cylinders in Figure 2-13

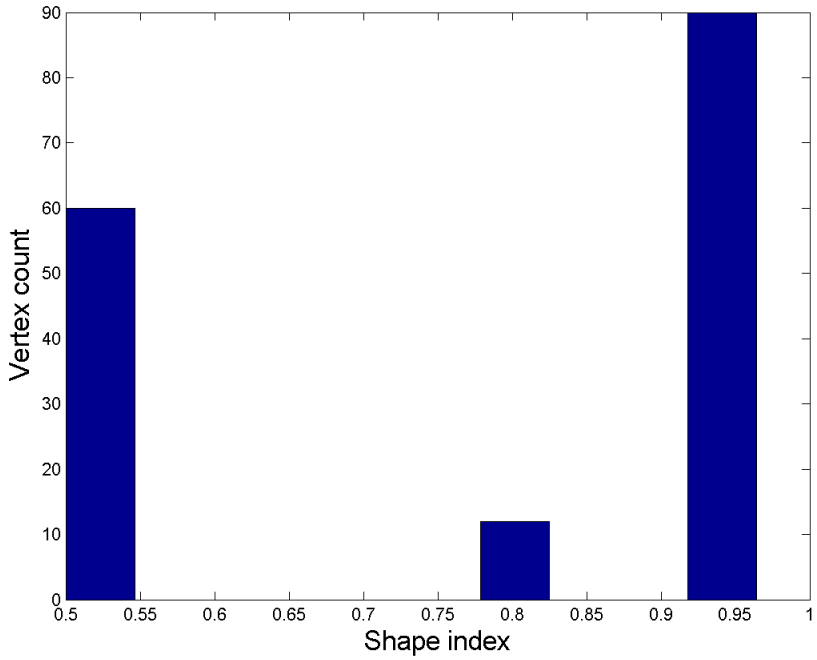


Figure 2-16: Distribution of S for the sphere in Figure 2-12(a).

The curvature calculations for the cylinder highlights the occasional instability of K when calculated using the technique described by Meyer et al. [2002]. A small number of large K values hides the fact that most of the values are within a reasonable range (Figure 2-13(c)) with a median of 5.78×10^{-4} compared to the mean of 1932.67. The other metrics are reasonably consistent when the vertex count is 2160 with S converging to 0.50 which matches the expected value for cylinders when using the mapping described by Koenderink [1990] and C changes with the values of k_1 and k_2. The distribution of k_2 for the cylinder (Figure2-15) demonstrates the stability of the k_2 metric for planar and almost planar regions. From these results we can see that the choice of meshing parameters and the type of surface has a strong influence on the numerical stability of the various curvature metrics. At every point on the cylinder in Figure 2-13(a) the value was 0.5 but the distribution of values for the sphere in Figure 2-12(a) ranges from 0.50 to 0.95 (Figure 2-16.)

To assess the robustness of curvature calculations in the presence of noise we applied a

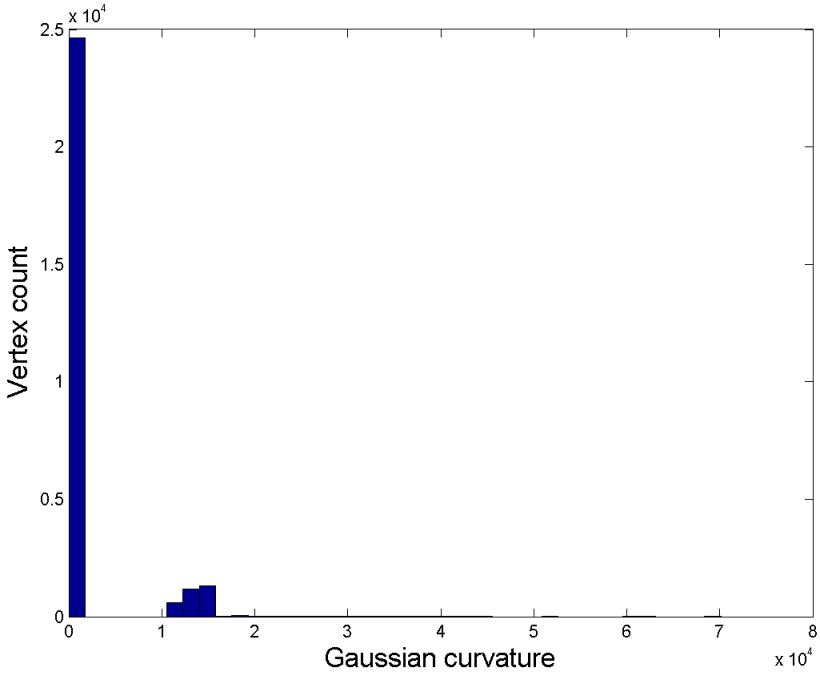
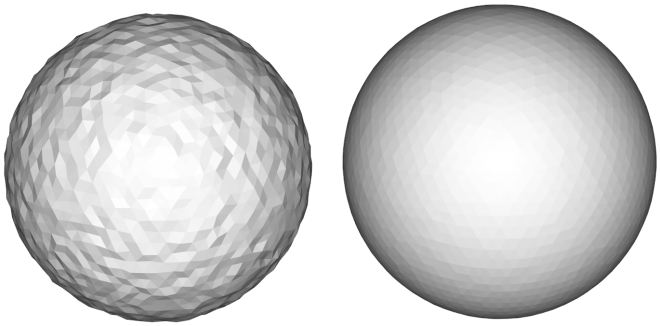


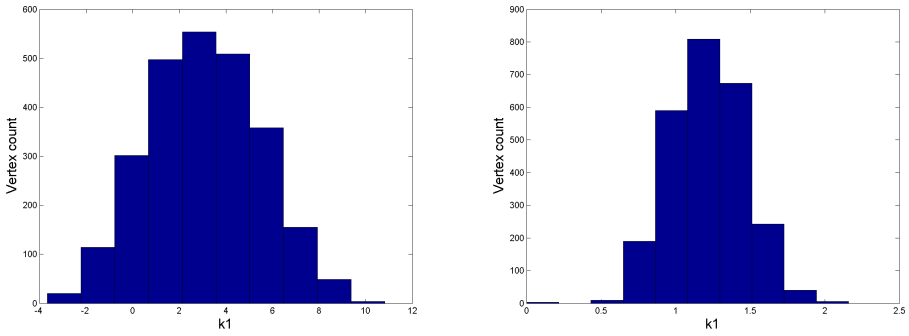
Figure 2-17: Distribution of K for the cylinder in Figure 2-13(c).

noise function is the sphere in Figure 2-12(b), plotted the distribution of k_1 and k_2 and compared this to the results of the unperturbed sphere. The noise function added a random value within ± 0.001 and ± 0.01 to the x,y and z coordinates of each vertex in the spheres shown in Figure 2-18. The distribution of k_1 for both surfaces after being perturbed (Figure 2-18) illustrates the high sensitivity of the technique described by Meyer et al. [2002] and strongly suggests that surfaces with high levels of noise will be difficult to process. These observations can also apply to models with varying levels of detail produced when meshing algorithms that adjust the number of vertices in regions of low and high curvature in order to make the representation more efficient and accurate.



(a) ± 0.01 (b) ± 0.001

Figure 2-18: Sphere composed of 2562 vertices with varying levels of noise within the specified range.



(a) ± 0.01 (b) ± 0.001

Figure 2-19: k_1 distributions for the spheres in Figure 2-18

2.3 Free-form Surface Segmentation

The segmentation of a free-form surface is the first step in many tasks in machine vision and related disciplines. The choices available for segmenting free-form surfaces are dependent upon the input data type. Many early techniques for segmenting free-form surfaces acquired using range scanners are natural extensions of image processing techniques as range data generated by laser time of flight scanners is represented as 2D image with pixel intensity indicating depth, referred to as 2.5D data. Other methods used to segment 2.5D data utilised curvature properties extracted from surfaces fitted to the range data to find regions and their boundaries. With the increased availability of computing power and decreasing cost of data acquisition devices, research began to focus more on 3D representations of acquired and CAD related data stored as point clouds and triangulated meshes.

2.3.1 2.5D Range Data Segmentation

Although much research into segmentation of free-form surfaces has changed focus from 2.5D to 3D datasets it is important to note that research into processing 2.5D data is still relevant as it is not always convenient or cost effective to acquire full 3D scans. Cost effective hardware solutions are readily available to the consumer for 2.5D scanning [Microsoft Corporation, 2011] and many researchers are utilizing these as a research platform [Kondori et al., 2011, Xia et al., 2011, Clark, 2011]. Techniques for segmenting range data fall into the following buckets.

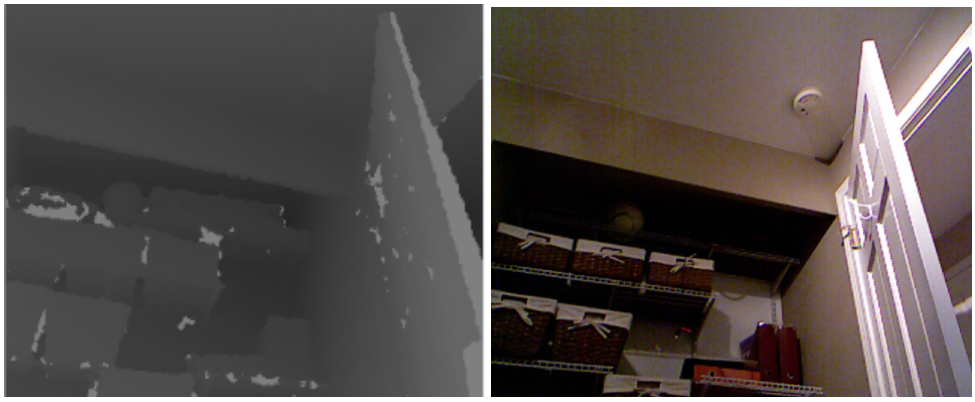
- Edge extraction
- Patch extraction
- Combined patch and boundary extraction

(a) Front (b) Side

Figure 2-20: Mesh generated from 2.5D range data [Moreno and A.Sanchez., 2004]

There are unique problems to processing 2.5D range data, the most significant is the self occlusion introduced into the data set because the scan is taken from a fixed location. Edges of different types have to be correctly identified when processing range data. Jump edges, crease edges and curvature edges are treated differently in the segmentation process as jump edges are caused by object occlusion, crease edges are a product of two or more surfaces meeting and curvature edges are more subtle changes in curvature where the surface normals do not change significantly but curvature measurements change [Jiang and Bunke, 1996]. Rendered 3D views of a surface generated from 2.5D range data is shown in Figure 2-20 and raw range and corresponding image data data taken from a Kinect [Microsoft Corporation, 2011] device is shown in Figure 2-21.

The main advantage of using edge based segmentation techniques is that they are more robust in the presence of noise [Coleman et al., 2010, Jiang and Bunke, 1996] compared to extracting regions from range data without pre-processing such as surface fitting, and there is a large corpus of existing literature on image segmentation that can be easily applied to range data [Mangan and Whitaker, 1999].



(a) Range (b) Image

Figure 2-21: Range and image data acquired from a Kinect device

Segmenting 2.5D range data using only edge detection techniques is not necessarily going to achieve the best possible results as these methods are relying on the presence of features that well define boundaries. In contrast the patch extraction techniques utilise a larger set of features in order to identify and segment regions.

The basis of many patch extraction techniques from 2.5D data are curvature metrics derived from surface fitting on the range data. Besl and Jain [1988] was the first to use H and K curvature signs to identify region types (Table 2.1). The curvature sign map output was then used as a starting point for fitting surfaces that would then form the segmented regions. The technique of using curvature signs and combining the signs with surface fitting formed the basis of many investigations focussed on segmenting 2.5D range data, 3D point clouds and triangulated meshes [Emanuele Trucco, 1995, Bose et al., 1996, Vieira and Shimada, 2005, Alrashdan et al., 2000, Yokoya and Levine, 1989].

Statistical learning techniques such as neural networks and self-organizing maps have been used with success to extract regions from range data. Koh et al. [1993] used SOMs of varying dimensions to build a hierarchical map of SOMs that segmented range images. The regions were generated using an error function that terminated the construction of more detailed SOMs, forming structures similar to quad-trees. Koh et al. [1995], Bhandarker et al. [1997] used a Hierarchical SOM (HSOM) to segment range images. Their

HSOM is organised into n square layers of SOMs. Each layer feeds into a smaller layer beneath it. For a square range image of width and height x the first layer of the HSOM will be a square of $\frac{x}{2}$, each successive layer having its dimensions halved. The HSOM uses coordinates of each point, its unit normal and its Gaussian and mean curvatures as the feature vector. The range image is fed into the HSOM and processed at every layer. The result of the segmentation is found by performing a breadth first search through the trees created by each layer and their links between each other. If the neuron represents the region closely then the neuron is marked as closed and is considered to be a segmented region. The range data and results of segmentation as the depth of the HSOM tree increases is shown in Figure 2-22.

Edge and patch detection techniques are a natural combination of features for identifying regions. Yokoya and Levine [1989] is one of the earliest examples of this approach. Similar techniques were used to extract regions with jump and fold edge detection combined with patch parametrisation to isolate and describe patches on 2.5D range data [Khalifa et al., 2000, Bose et al., 1996, Vieira and Shimada, 2005].

A recent implementation of combining patch and edge data by Benlamri and Al-Marzooqi [2004] used piecewise hermite interpolation surfaces (PHIS) to approximate regions on the triangulated mesh generated from range data. The mesh is segmented by identifying edges and the type of edge (jump boundary, crease and smooth.) Using surface normals a dense triangulated mesh is generated and PHIS are generated and blended to create the regions. Region merging across boundaries only happens where the edges between regions are smooth. The mesh is coarsened as the surfaces are fitted and merged. Within bound regions are many PHIS that are merged into each other. An example of this approach is shown in Figure 2-23.

Machine learning techniques combined with the features H, K, depth and surface normal were used in a relaxation oscillator network for segmenting range images [Liu and Wang, 1999]. A relaxation oscillator consists of an excitatory and inhibitory unit connected to each other in a feedback loop, the network requires one oscillator per pixel in the range

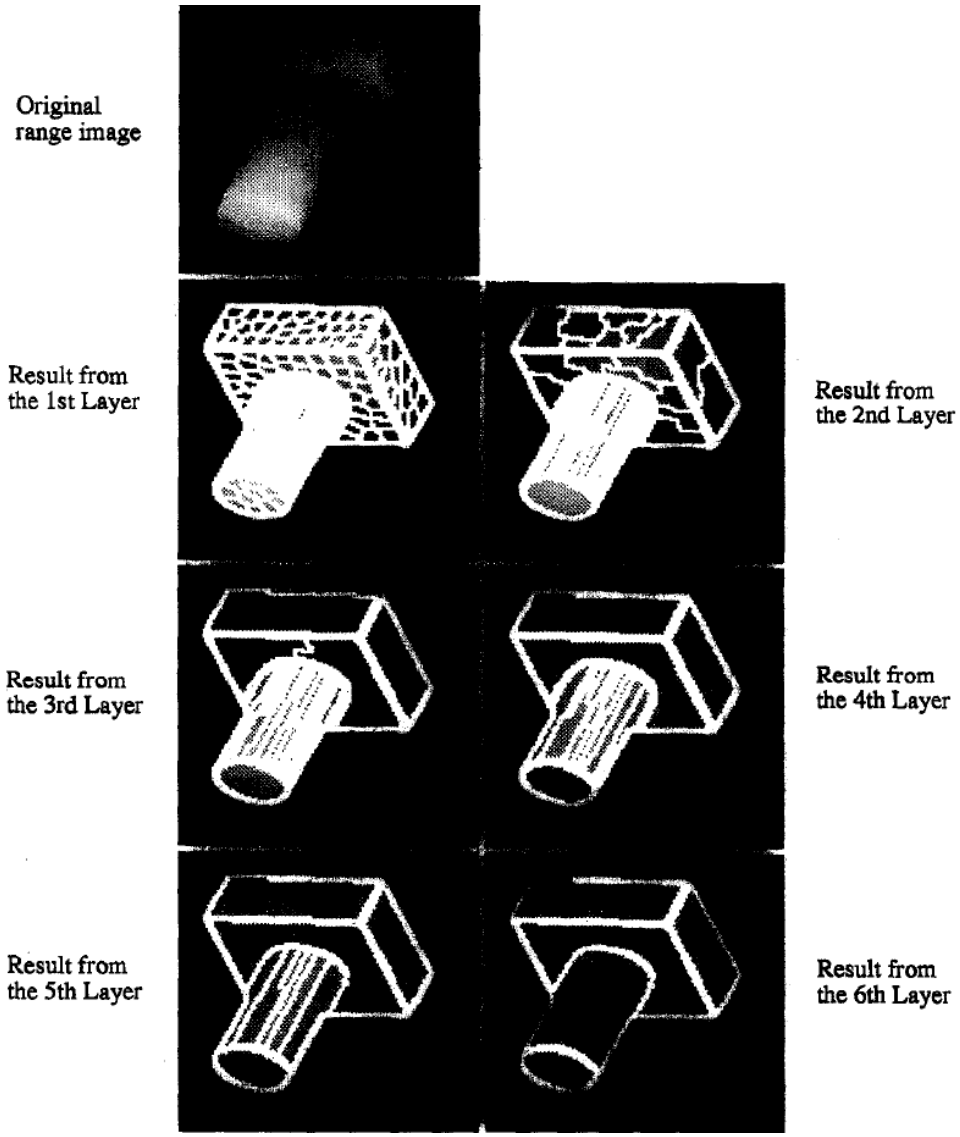


Figure 2-22: Original range data and segmentation results at each level of the HSOM reprinted from Bhandarker et al. [1997] ©1997 with permission from Elsevier

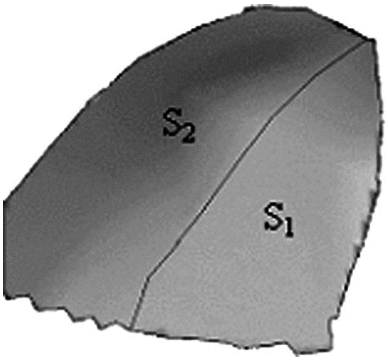


Figure 2-23: Example of smooth edge where surfaces will be blended reprinted from Benlamri and Al-Marzooqi [2004] ©2004 with permission from Elsevier

image. As the range data is processed oscillators that are in phase sync when stimulated by input data representing like pixels. The inhibitive component of the oscillator network actively desynchronises oscillators that are out of phase to accentuate the differences. Oscillators that represent the background stop oscillating after a short period of time as they will not synchronise with others to reinforce the grouping. No prior knowledge is required and the emergent behaviour of the oscillators provides the segmentation. Their LEGION system requires many calculations to model the oscillators and requires the user to set parameters manually. It is a computationally intensive model that is not fully automated. Dissimilarity represented by weak lateral connections forms the boundary between pixels and strong lateral connectivity shows similarity and forms the patch. It's a computationally intensive model as the modelling of a cubic function is required for each oscillator over time as the range data is processed.

Learning methods have been applied to identify particular patch types and their approximate parameters. Chella and Pirrone [2002] used a Self Organizing Map to segment range data and then a neural network was used to estimate the parameters of the surface patch represented by each cluster in the SOM. Alrashdan et al. [2000] used a Laplacian filter and surface normals passed into a Self-Organizing map to detect step and roof edge points. The H K sign map was fed into neural network to extract regions. The results were combined to produce segmented regions.

(a) Model (b) Watershed (c) Spectral clustering

Figure 2-24: Comparison between segmentation methods by Liu and Zhang [2004] ©IEEE

2.3.2 3D Model Segmentation

Hardware to extract 3D model data and software that is able to recreate 3D data from multiple 2.5D views has become more commonplace. This has driven the creation of new and the extension of existing segmentation techniques previously used on 2.5D range data to segment 3D data.

Mangan and Whitaker [1999] applied watershed segmentation techniques developed for 2.5D image segmentation to 3D object segmentation. Convex regions are extracted using local minima and gradient descent using the total curvature ($D = 4H^2 - 2K^2$). The watershed segmentation model requires the setting of thresholds for preprocessing to remove noise. Razdan and Bae [2003a] refined this method by inserting vertices where the sum of two connected vertex normals are greater than a threshold value. Pan et al. [2004] took a similar approach, using a surface flatness metric based on area and the unit normals of the surrounding vertices. Edge detection using dihedral angle and watershed segmentation has been successfully applied to both CAD models and laser scanned objects [Razdan and Bae, 2003b].

Watershed segmentation is prone to over segmentation and can fail on simple shapes such as a simple L shaped object [Liu and Zhang, 2004] illustrated in Figure 2-24. The output of these two segmentation algorithms illustrates the problem where the expecta-

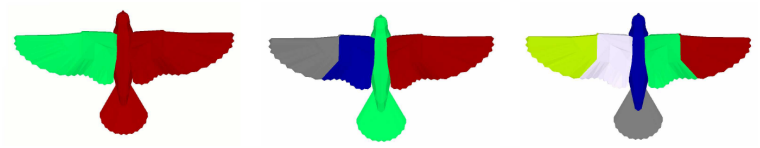


Figure 2-25: Results of mesh decomposition for increasing values of n taken from Katz and Tal [2003] ©2003 Association for Computing Machinery, Inc. Reprinted by permission.

tion of how an object should be segmented doesn't always match. The L shaped object would be considered a single region if the intent was to recognize letters, but if the intent was to find the shape primitives such as rectangles, cubes and cylinders in a scene then neither of the results are sufficient. Domain specific techniques such as statistical shape modeling and spatiotemporal analysis have been successfully applied to segmentation of biomedical data [Zhang et al., 2010, Munsell et al., 2008] and image sequences [Apostoloff and Fitzgibbon, 2006, Vincent et al., 2000]. There are no techniques that could be universally applied to all segmentation problems since the definition of a perfectly segmented object is driven by context, human perception and the problem being solved.

Katz and Tal [2003] addresses some of the shortcomings of watershed segmentation for 2-manifold meshes such as reliance on the quality of triangulation and low magnitude convex regions causing over segmentation. The use fuzzy decomposition to divide the model into n parts by taking the number of components required as a parameter and using the geodesic distance and weighted dihedral angle between faces to calculate the distance between all faces. The n faces with the greatest distance between them are chosen as the representative faces for each component of the model. A minimizing function is used to assign faces to patches and the regions at the boundary of each patch are partitioned using a flow network. This process repeats until one of four different stopping conditions is met, all using manually set values. The results of decomposing the model with increasing values of n is shown in Figure 2-25. Hierarchical methods were further extended by automatically generating too many seeds and merging regions based on a cost function [Lai et al., 2009].A sample segmentation is shown in Figure 2-26.

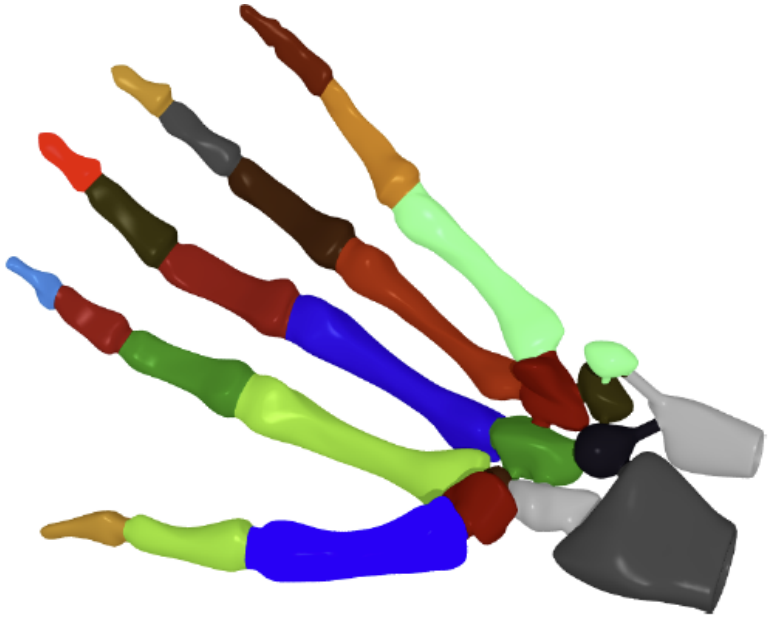


Figure 2-26: Improved hierarchical segmentation using random walks reprinted from Lai et al. [2009] ©2009 with permission from Elsevier

Investigation into machine learning techniques for segmentation have continued into the 3D object domain with methods such as spectral clustering to segment triangulated surfaces [Liu and Zhang, 2004] which addressed some of the weaknesses of the watershed segmentation technique (Figure 2-24). This system requires the manual setting of the number of eigenvectors to use which determines how many clusters the K-means algorithm will expect to find. Sidi et al. [2011] extended spectral clustering to learn from objects that belonged to the same class, e.g. sets of chairs and identified pairwise similarities between objects to remove the reliance on setting a manual parameter. Park et al. [2003] starts with a point cloud to represent a 3D model and applies k-means clustering to define the regions for each patch to cover. A manual estimate of the number of clusters is required.

2.4 3D Model Indexing and Retrieval

Research into 3D model indexing and retrieval has concentrated on object matching and are either based on models that contain a volume or are free-form range images with or without occlusion affecting the completeness of the dataset. Some of the techniques described in this review aren't directly applicable to free-form surfaces but are still valid sources of research inspiration when evaluating different representations, indexing and recognition techniques.

Graph comparison techniques extract structure from 3D models and represent the relations between different components of the model as a graph. The two most common graph representations of 3D models for indexing and retrieval purposes are Reeb and skeletal. Skeletal graphs require manifold 3D models and are built using a voxelization of the volume followed by thinning [Tangelder and Veltkamp, 2004, Sundar et al., 2003]. Free-form surfaces aren't always going to have volume, so this method isn't relevant to our research.

Reeb graphs use a μ function [Tung and Schmitt, 2005, Biasotti, 2004] to track changes in shape topology. Examples of μ functions are height and geodesic distance. An example of an object decomposed into a Reeb graph is shown in Figure 2-27. Rotation invariant functions such as geodesic distance are preferred when using Reeb graphs for 3D model processing to avoid any issues with finding consistent model orientation. To improve the performance of the geodesic distance μ function the distance measurements are made between patches that are generated via resampling of the surface of the object [Hilaga et al., 2001]. The graph is constructed by partitioning the μ function into arbitrarily sized buckets and generating graphs with edges connecting regions that exist in the same μ value range. Multi-resultion Reeb graphs subdivide regions to allow for matching between regions at different levels of subdivisions.

Spectral graphs are used to index solid models by calculating the eigenvalues of the graph adjacency matrix to index and retrieve solid models. Operations on graphs can be expen-

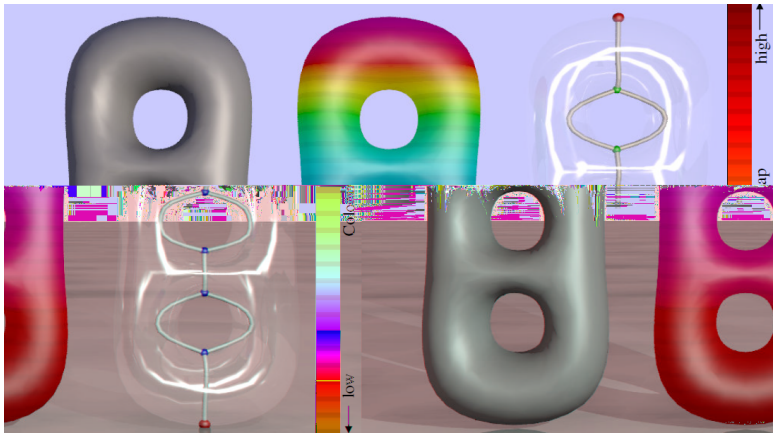


Figure 2-27: 3D model decomposition into Reeb graph and with distance function scale by Pascucci et al. [2007] ©2007 Association for Computing Machinery, Inc. Reprinted by permission.

sive. Calculating graph edit distance is NP hard [Tangelder and Veltkamp, 2004], making graph representations unattractive for efficient indexing and retrieval. B-Rep (boundary representation) graphs are used to encode and store CAD models. The most common formats for encoding B-Reps are IGES and STEP. The vertices of the graph are surfaces and the edges describe connectivity between these surfaces [Iyer et al., 2005].

Geometric hashing is a storage intensive technique that allows for approximate surface matching by taking features (such as surface normals) then encoding as triplets of features with their spatial relationships in a scale invariant manner [Wolfson and Rigoutsos, 1997]. This scheme has been used to find partial matches and similarity between 3D models in a database. To reduce the amount of space required to index the models, only salient features are encoded in the geometric hash map. The saliency of a local feature on the mesh is determined by its variance and magnitude of curvature compared to neighbouring regions [Gal and Cohen-Or, 2006]. An improved approach was taken by Funkhouser and Shilane [2006] where both similarity between the features and geometric deformation are used to create a priority queue of potential matches that are then processed with a comprehensive matching algorithm.

Breaking down surfaces into numerical representations allows for simpler indexing, retrieval and comparison techniques to be utilised. Research into the comparison of distributions is extensive and there are many techniques that can be applied to comparison problems.

Shape Distributions is the collective name for distributions calculated using functions that describe the object as a whole. The motivation for using these distributions is to reduce the comparison process to histogram matching [Osada et al., 2002] to identify classes of shapes. Examples of shape distribution functions are:

- A3: Angle between three points selected at random
- D2: Distance between random point pairs
- D3: Area enclosed by points described in A3

Vandeborre et al. [2002] combined the following shape descriptors to generate descriptors in a similar fashion to shape distributions for a 3D volume:

- Histogram of the curvature index C [Koenderink and Van Doorn, 1992] for the entire object
- Distance index D2 taken from Osada et al. [2002] for N random pairs on the surface of the 3D model
- Volume distribution using the methods described in Zhang and Chen [2001] to calculate a Fourier transformation of the 3D volume.

This descriptor moderately successfully recognizes some classes of objects from the 3D CAFE data set [Titus Zaharia, 2000.] compared to using the components of the compound descriptor individually.

Curvature maps [Gatzke et al., 2005] use geodesic fans or rings to accumulate mean or Gaussian curvature as a function of distance. This descriptor was used to identify visually similar regions on a 3D model with irregular meshing using both the ring and fan based accumulator functions with a combination of mean and the square root of Gaussian curvature.

Shape Histogram functions use a region binning function such as shell or sector bins centred at the centre of mass and aligned using the principal axes transform to break down the shape into a set of bins and a weighted Euclidean distance function to calculate the distance between two shapes [Ankerst et al., 1999, Iyer et al., 2005].

Extended Gaussian Images (EGI) creates a unique image for convex 3D models that is a projection of surface normals onto a sphere that can be transformed into a histogram that can then be used to compare surfaces [Horn, 1983, Kazhdan, 2004]. Non-convex models are handled by extracting their convex components and processing them separately. The histogram representation is compact and simple to compare with others, but is sensitive to transformations and noise. The COSMOS representations scheme [Dorai and Jain, 1997a] projects a segmented 3D model acquired from range data onto a sphere with each point representing a region that shares the same shape index. Fu et al. [2008] projects the integral of Gaussian curvature of a free-form surface onto a 2D spherical coordinate to provide a scale and transform independent descriptor.

Spin Images are an object recognition technique that work in scenes with multiple objects that are subject to occlusion [Johnson and Hebert, 1999]. They generate 2D histograms that are centered at each vertex and oriented using the surface normal. The magnitude of the histogram is the number of projected points through that point in the plane. To generate sufficient spin images for searching for objects in cluttered scenes the 2D plane of the image is rotated through 360° by a number of increments. This generates a huge number of images per object so in an attempt to reduce the search space PCA is used to compress the representation into a relatively manageable size. This technique does require the setting of 2D histogram bin sizes and the support angle (compensates for self

occlusion at the cost of level of detail). Clustering has been used to further reduce the space required to represent each of the representative spin images [Assfalg et al., 2007].

Spherical harmonics are used as a rotation invariant descriptor for manifold 3D surfaces derived from projecting the model onto a sphere using various transformations such as ray length and voxelization [Vranic et al., 2001, Vranic and Saupe, 2002]. Funkhouser et al. [2003] built a representation using a voxel representation of the 3D model and generating shells of concentric spheres, marking the sphere where a voxel is present, to create a spherical mapping. The 2D feature descriptor for the shape is created from the magnitude of the first n Fourier harmonics coefficients for each of these shells. This has been successfully used to index and retrieve whole manifold surfaces. Since orientation is not a part of the descriptor each shell could be rotated at any angle independently, so there is a possibility for a model that has been sliced and rotated about the central axis.

Moments in 3D are an extension of moments used in 2D image recognition and gives us the volume, orientation and centre of mass of the object and a signature that is invariant to orientation, position and scaling [Sadjadi and Hall, 1980]. Saupe and Vranic [2001] demonstrated that spherical harmonics performed better than moments based on their experiments.

View based indexing and retrieval techniques use multiple 2D rendered views of a 3D model to construct an index based on the image features of the model. Ohbuchi et al. [2010] uses a bag of visual words approach, using SIFT to extract features from the image and a pre-computed codebook to perform vector quantization on the words which is then transformed into a histogram. Mhamdi et al. [2010] attempts to reduce the number of views required to build a representative set of features by using PCA to align the 3D model. A survey of the current view based indexing and retrieval techniques for 3D models [Petre et al., 2010] found that increasing the number of views didn't necessarily give better retrieval results and that the MPEG-7 curvature scale space descriptor was better than both Zernike moments and the MPEG-7 2D angular radian transform. This approach to the indexing and retrieval problem where there is a loss of spatial relation-

ships in the encoding has been successfully applied to textual analysis [M. Sahami and Horvitz, 1998, Blei et al., 2002], and object and image retrieval [Ohbuchi et al., 2010, Rubner et al., 2000, Gao et al., 2010, Zheng and Gao, 2008]. In these cases scale and/or spatial relationships have been removed from the object descriptor and still been successful in achieving their desired outcomes.

Techniques used to find approximate matches in range data and 3D models tends to focus on identifying complete objects in a scene [Fan et al., 1989, Johnson and Hebert, 1999, Zhang and Hebert, 1999] whereas we are interested in identifying parts in a database of models that contain regions similar to the search criteria. Representations that rely on alignment such as spherical harmonics, view based indexing and extended Gaussian images are not as useful when searching for sub-matches within a free-form surface as the required alignment isn't always known ahead of time, and matches can occur at different scales and locations.

2.5 Conclusion

In this Chapter we examined the SOM, 2.5D and 3D data segmentation and indexing and retrieval techniques. The SOM was identified as a flexible machine learning technique that has been applied to a broad range of problems ranging from compression to handwriting recognition. It has been shown to be the basis of effective segmentation techniques of 2.5D and 3D range data [Chella and Pirrone, 2002, Koh et al., 1995, Bhandarker et al., 1997] and the potential for new curvature measurement combinations to be used to describe regions on a free-form surface makes it an ideal basis for further research.

Curvature is an intuitive way to describe the shape of a free-form surface and what makes it unique, or similar to others. The identification of reliable and simple discrete curvature measurement techniques [Meyer et al., 2002] for surfaces represented as triangulated meshes is essential to our research as we require a simple and effective algo-

rithm to measure curvature that works on one of the simplest representations of free-form surfaces. Curvature metrics have been used to identify the properties of objects in many investigations into segmentation and indexing and retrieval [Besl and Jain, 1988, Emanuele Trucco, 1995, Bose et al., 1996, Vieira and Shimada, 2005, Alrashdan et al., 2000, Yokoya and Levine, 1989, Koh et al., 1993, 1995, Bhandarker et al., 1997, Liu and Wang, 1999].

Our evaluation of segmentation techniques is an insight into the existing methods available and identify new branches of research. We identified current techniques such as hierarchical decomposition and previous research using machine learning techniques to segment 2.5D and 3D representations of objects. The applications of machine learning and clustering techniques to identify the components of different models exists throughout the literature [Liu and Zhang, 2004, Tangelder and Veltkamp, 2004, Sidi et al., 2011, Park et al., 2003] and we have identified a gap where not all combinations of the curvature metrics we reviewed have been evaluated.

We gave an overview of the current technique for performing this task for 3D models and found that most of these systems are concerned with seeking objects within a scene [Fan et al., 1989, Johnson and Hebert, 1999, Zhang and Hebert, 1999] or matching of complete 3D models within a database [Ohbuchi et al., 2010, Funkhouser et al., 2003]. We discussed the successful application of distribution based descriptors where scale, and in some cases spatial relationships, have been removed from the object descriptor such as proposed by Osada et al. [2002] and Vandeborre et al. [2002] and will be using this bag of words approach to the problem of indexing and retrieval of whole and partial 3D models.

Chapter 3

Using Self-Organizing Maps to Classify Vertices

The extraction of basic information about points on a free-form surface is the first step in performing tasks such as segmentation and indexing. We need to find properties that can be used to distinguish from and find similarities between regions on the surface. Examples of free-form surfaces are shown in Figure 3-1 and one represented as a triangulated mesh in Figure 3-2.

Curvature metrics describe how the surface bends around a point on a surface. We will use a number of curvature metrics and calculation techniques for discrete data, described in Section 2.2, as input into a Self-Organizing Map used to classify vertices on a free-form surface. The surface can be described using the SOM as a clustering function and the underlying structure of the triangulated mesh to describe the connectivity of the points.

Besl and Jain [1988] used simple rules to determine surface types by using pairs of curvature metrics and their signs. The disadvantage of their technique is that is is unable to represent varying degrees of the same curvature type based upon their set of rules, e.g. their technique is unable to distinguish between two regions that are classified as ridges ($k_1 < 0$ and $k_2 = 0$) where one region's k_1 value is far smaller than than the others which would mean one ridge had a gentle slope and the other a much sharper slope.

We will focus on the following parameters during our investigation:

- SOM Size
- Features
- Training data set
- Training iterations

The SOM size was chosen as one parameter to vary as it determines how many vertex types could potentially be identified, each curvature descriptor identifies different properties of a given vertex so each combination will group vertices differently. The training

data set determines how sensitive the SOM will be to feature combinations. If a SOM is trained on only a single object class it will be insensitive to others. Training iterations determine how many fine and coarse training iterations will be executed, which determines how finely tuned the SOM will be to input data. If a SOM is over trained it will be too sensitive to variations in features and produce noisy results, if under trained it won't be able to distinguish between different classes of input data.

Both the coarse and fine training processes cause the SOM to alter its weights to more closely match the stimuli with the neighbourhood of affected neurons centered at the best matching unit (BMU). The coarse training process starts by heavily affecting the weights of all of the neurons that make up the SOM with a decay function that gradually reduces the scope and strength over time. The fine training process has an effect on the weights of neurons within a small neighbourhood. The update and neighbourhood functions are detailed in Equations 2.5 and 2.4 on page 14.

We will investigate what effect variations in experimental parameters have on the SOM output when classifying vertices by using a set of surfaces that represent conic sections (paraboloids) as training data and as surfaces whose vertices will be classified. Conic sections are being used as they represent a diverse number of curvature types and of varying magnitude within a relatively compact set of shapes. These surfaces were created using the mathematical functions described in Table 3.1 and illustrated in Figure 3-1. These surface types have been used in the past to segment objects into regions from range data [Visintini et al., 2006] using region growing and surface fitting.

The triangulated meshes were generated by using the function that describes each surface to create a parameterised surface and then meshed using GMSH [Geuzaine and Remacle, 2003] to create a mesh with uniformly distributed edges and vertices illustrated in Figure 3-2. The vertex, face and edge counts for each of the meshes is shown in Table 3.2.

To manage the results of these experiments we used the OpenJPA Object-relational mapping implementation [Foundation, 2011c] to store the results in a database by annotating

Table 3.1: Parametric equations for paraboloids

	Parabolid	Elliptic	Hyperbolic
$x(u, v)$	$\sqrt{u}\cos v$	$\sqrt{(u/2)} * \cos(v)$	u
$y(u, v)$	$2\sqrt{u}\sin v$	$\sqrt{(u/2)} * \sin(v)$	v
$z(u, v)$	u	u	uv
u range	$0 \leq u \leq 10$	$0 \leq u \leq 4$	$-3 \leq u \leq 3$
v range	$0 \leq v \leq 2\pi$	$0 \leq v \leq 2\pi$	$-3 \leq v \leq 3$

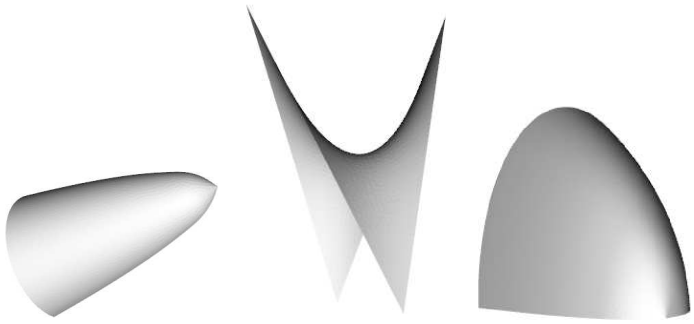


Figure 3-1: Solid rendered images of Paraboloid, Hyperbolic paraboloid and Elliptic paraboloid [MacLennan and West, 2008].

source code with additional properties that are used by OpenJPA at runtime to instrument the Java bytecode with database queries. Using a database to store our results and perform post-experimental analysis allowed us to focus on our research instead of the less interesting problem of storing experimental data and allows us to query the object data using SQL queries. The results were visualised using our own colour to vertex mapping algorithm and the standard OOGL file format [Amenta et al., 1995] combined with the GUI widgets provided by the Visualization Toolkit (VTK) [Schroeder et al., 2006] to display rendered 3D models. We used the Apache commons library [Foundation, 2011a] to perform statistical analysis of our results and data sets and Matlab or Excel to plot summary data.

To visualise the cluster membership of each vertex we will use a colour mapping algorithm for vertex colour. The colour mapping works by iterating through the (x, y) BMU

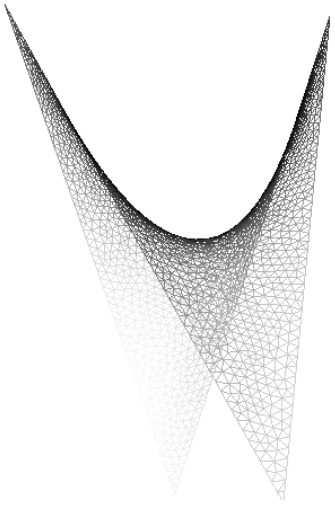
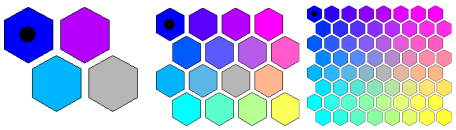


Figure 3-2: Hyperbolic paraboloid rendered as a wireframe.

co-ordinates of the SOM and incrementing values of red, green and blue, to generate colours and storing the colour values for each BMU in a hash table. Examples of this mapping are shown in Figure 3-3 with the origin BMU $(0, 0)$ in the top left hand corner.

To ensure repeatability of our experiments we always linearly initialise the neuron weights across the x and y dimensions of the SOM. In some cases the curvature values are found to be $\pm\infty$ so we ignore these values when using Min-Max linear initialization that maps the values to an interval of $[0, 1]$ and when calculating the Euclidean distance between feature vectors and neurons in the SOM. With sufficient training iterations it has been shown that even with a random initialization the SOM will converge on a solution even for very large ranges of input data [Kohonen, 2001, Page 312]. The benefit of analytical techniques such as PCA based initialization is that the SOM will converge on its stable state at a faster rate because some of the ordering has been taken care of before the training process so the SOM can spend more time on fine-tuning the representation [Vesanto et al., 1999a].

The distribution of colours associated with vertices on the face gives a visual indication of the quality of the classifier. A surface with very little variation in colour suggests that the SOM is not sensitive enough to its features when clear variations in curvature are not



(a) 2×2 (b) 4×4 (c) 8×8

Figure 3-3: Examples of colour mapping tables used to visualise BMUs on a surface who's vertices have been clustered using a SOM.

Table 3.2: Surface statistics for Paraboloid mesh representations

	Parabolid	Elliptic	Hyperbolic
Edges	7875	21283	4668
Vertices	2686	7170	1637
Faces	5190	14114	3032

reflected in vertex colour changes. In other cases there may be a lot of variation of colour on the surface which could indicate that the mesh representing the surface was built from noisy data or the SOM is too sensitive to changes in curvature. The SOM error metric is labelled Q and the function used to calculate it is detailed in Section 2.6.

3.1 Paraboloids

In this Section we will use the vertices that make up the paraboloids as both training data and features to classify to determine how the SOM size, feature set, training iterations and paraboloid data set interact to create a classifier used to build surface patches. Throughout the experiments we will refer to the quantisation error metric Q (Equation 2.6 on page 18) to measure how well the SOM's clusters represent the vertices that make up the mesh representation of the paraboloid data set. These paraboloids were created using implicit functions initially stored as IGES files [Association, 1999] which describe the surfaces as functions, converted to meshes using GiD [CIMME, 2007] and then imported as objects into our database backed object storage.

3.1.1 SOM Size

The maximum number of clusters that can be represented by a SOM is determined by the number of neurons that constitute the map e.g. a SOM made up of (2×2) neurons can represent 4 different clusters. It is important to choose an appropriate SOM size to avoid over and under fitting data to the clusters represented by the neurons that make up the SOM. Too many neurons will cause regions of similar curvature on the surface to be associated with many different neurons and will appear as noise, too few will group dissimilar vertices together.

To measure how the SOM size affects the training and classification process we measure the mean quantisation error Q (Equation 2.6) for all vertices of the 3D models illustrated in Figure 3-1. To identify how many neurons are used to describe we plot the percentage of neurons fired when the SOM is used to classify vertices. Images of the 3D model are shown with the vertex cluster membership associated with a colour to visually evaluate the results of classifying vertices. We will also use the U-matrix (Figure 2-5) annotated with the proportion of SOM hits to illustrate the distribution and relative size of each cluster membership.

We fixed the number of coarse training iterations to 10 times the number of vertices in the training data set (10 epochs) and the fine training iterations to 0 (0 epochs) and used the features $K\ H\ C$ and S while varying the SOM size. The fixed parameters were chosen as they were shown to be a good starting point for a similar experiment detailed in MacLennan and West [2008].

The Self-Organizing properties of this learning technique is illustrated in Figure 3-4 where the mean quantisation error (Q) shows little variation when it grows beyond 8×8. We executed a single training and classification process for each SOM size to generate these results as the linear initialization process guarantees that the results will be the same when experimental properties are left unchanged. When the SOM size is 2×2 the Q error is at its peak and then drops sharply when the size is increased to 4×4. The increased

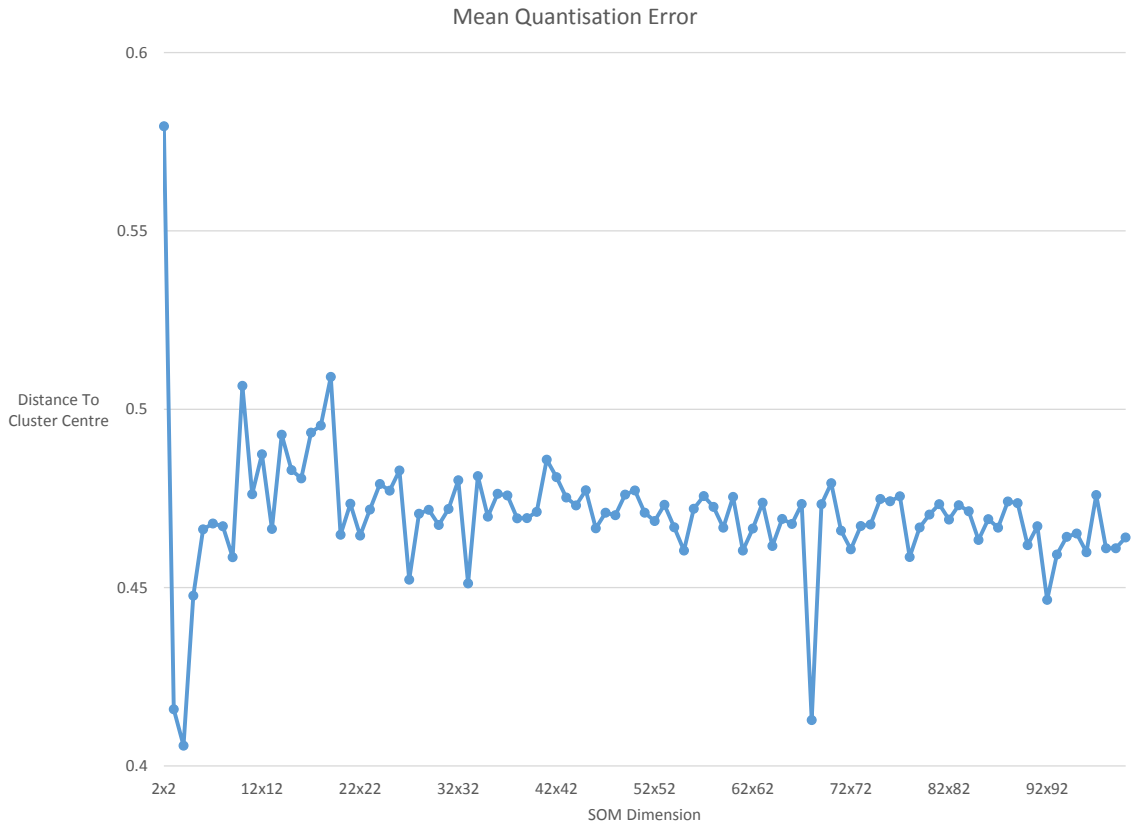
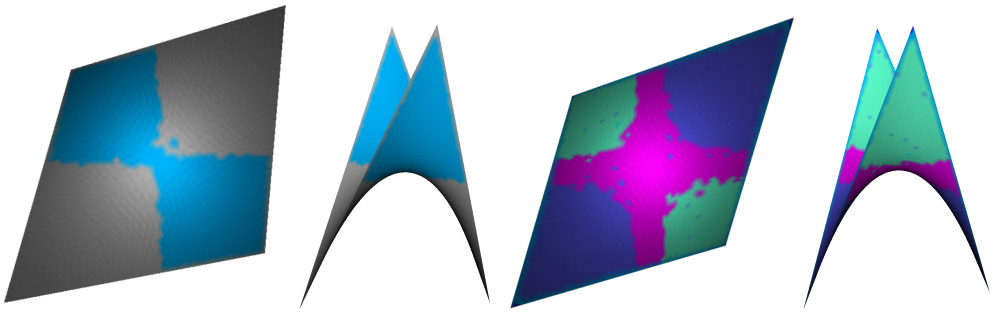


Figure 3-4: Plot of mean Q error for surfaces being classified by a SOM trained on exemplars

number of clusters that can be represented using the SOM neurons allows for a more accurate representation of the various curvature types. This effect is illustrated in Figure 3-5 which shows a top view of the hyperbolic paraboloid with the vertices coloured by their BMU. Note the lack of variation in colour for the 2×2 view where the only distinction made between the regions is whether the curvature around a vertex is convex (blue) or concave (grey) whereas the 4×4 SOM can distinguish between convex (green), concave (dark blue) and transitioning between concave and convex with curvature in both directions (purple). There are a small number of artefacts in the output due to noise introduced by the meshing process that converts a function into a triangulated mesh and relative instability of some features when the surface is flat or close to flat.



(a) 2×2 (Top) (b) 2×2 (Side) (c) 4×4 (Top) (d) 4×4 (Side)

Figure 3-5: View of Hyperbolic paraboloids with vertices coloured by BMU

There appears to be little benefit in increasing the SOM size once it is greater than 4×4 for the feature set $K\ H\ C$ and S as the Q error maintains a steady level beyond that point. To understand why there appears to be a diminishing return on the increased size of the SOM we plotted the proportion of neurons fired when classifying the vertices contained in the exemplar training set (Figure 3-6). As the SOM increased in size the proportion of neurons fired when the training data was classified trended downwards in an almost linear fashion. This relationship changes depending upon the feature set and training data as illustrated by the results in Figures 3-23 and 3-24 on pages 81 and 82. In this case the number of neurons in the SOM begins to exceed the number of clusters naturally occurring in the training data set. The number of clusters for a surface is due optimizations in the meshing process that affects the range and interval of curvature values.

We also visualised the distribution of neurons being fired for the 2×2, 4×4, 10×10 and 30×30 SOMs and their U-matrices (Figure 3-7) to understand how it changes as the SOM increases in size and the difference between two SOMs that have similar Q error but different percentages of neurons fired.

The U-matrices show the feature distance (Euclidean) between the neurons using shades of grey. As the distance increases the regions between adjacent vertices get darker. The distribution of neural stimulation as the training data set is classified is marked in red

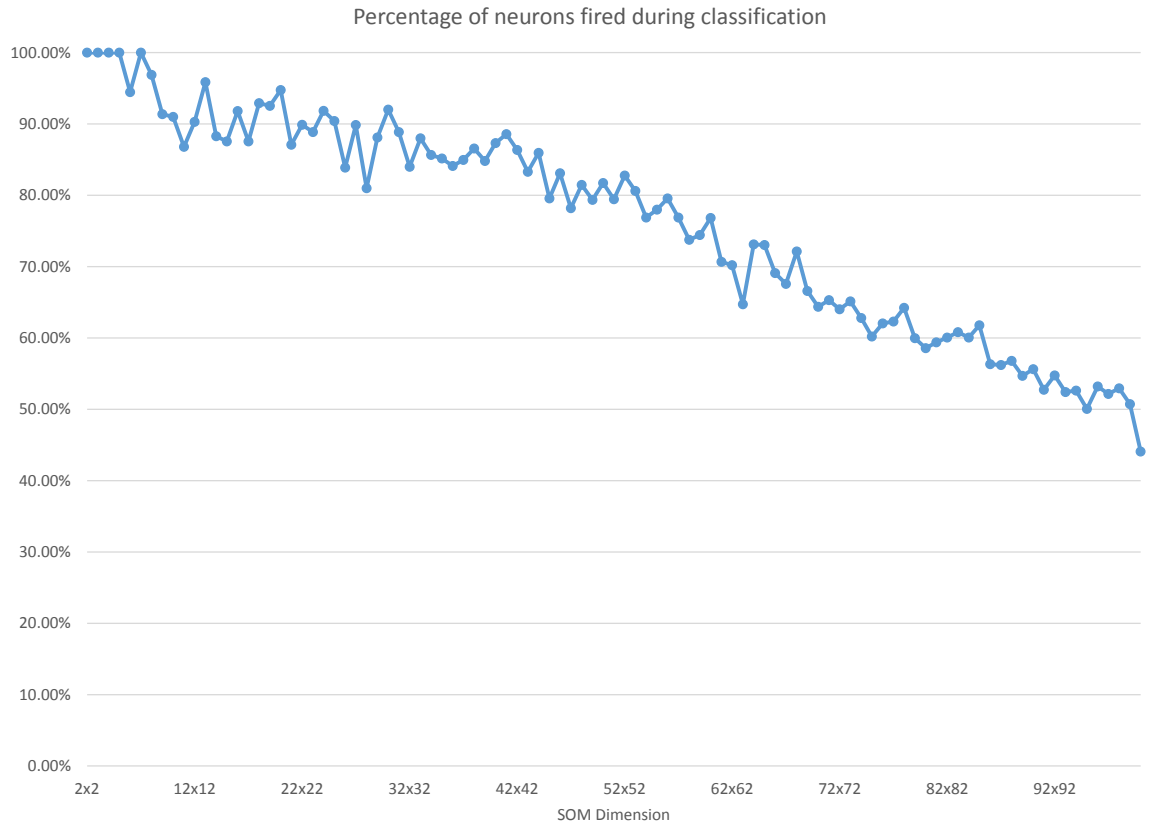
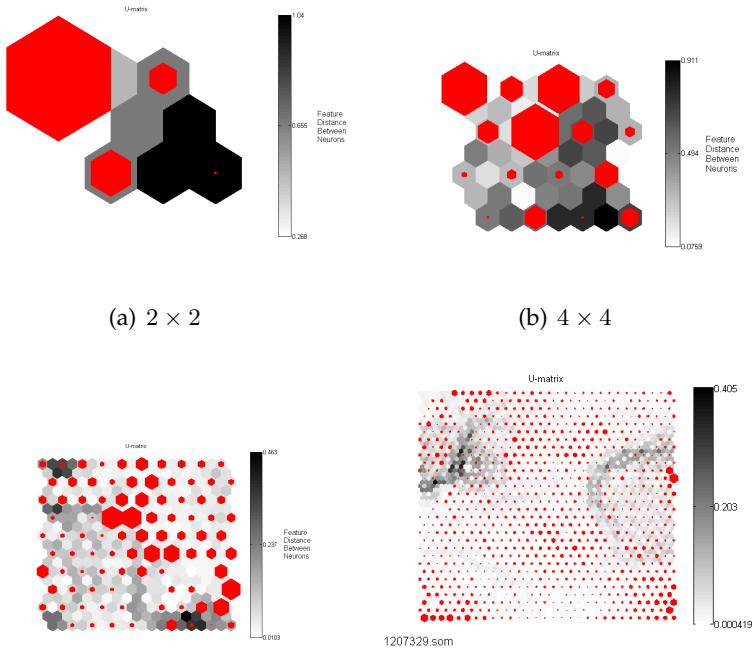


Figure 3-6: Plot of percentage of neurons fired by exemplar data set when classified with SOM trained on the exemplar set in Figure 3-1 on page 56

and the larger the hexagon the higher the proportion of stimulation associated with the neuron. The 2×2 SOM U-matrix has a very large proportion of the neural stimulation centred on a single neuron with very large distances, indicated by the dark grey and black between each of the neurons. The much larger 30×30 SOM shows where there is very little feature distance between most of the neurons (note the smaller scale in the white-black gradient bar) and a relatively even distribution of neural stimulation with some neurons not showing any hits. The 4×4 U-matrix shows both many neurons that have a large distance between each other with no single neuron dominating the distribution and all neurons firing during the classification process. The 10×10 U-matrix shows attributes that are similar to the 30×30 U-matrix where the feature distance between neurons is less than that of the 4×4 U-matrix and not all neurons firing during the classification process.



(a) 2×2 (b) 4×4

(c) 10×10 (d) 30×30

Figure 3-7: BMU hit distribution for the exemplar training data set

The lightly coloured regions in the U-matrices of the larger SOMs represent a small feature distance between neighbouring neurons which is a symptom of a SOM that is too large. Ideally neurons represent distinct sets of features that are quite different to other neurons, but in the case of larger SOMs there are many neurons that are very similar to neighbours which will introduce noise from over-fitting data. The effect of having an oversized SOM is also illustrated by the stability of Q error as the number of neurons that fire decreases as the SOM size increases. Similar regions that were associated with a single neuron for smaller SOMs are represented by larger regions of neurons with small feature distances as demonstrated by the U-matrix 30×30 in Figure 3-7.

3.1.2 Sigmoid function

Throughout the thesis we plan to use the naïve approach of linearly scaling the feature values to fit within the range of $[0, 1]$. To validate this approach we investigated the im-

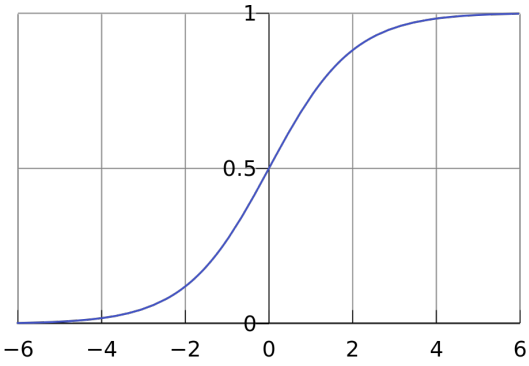


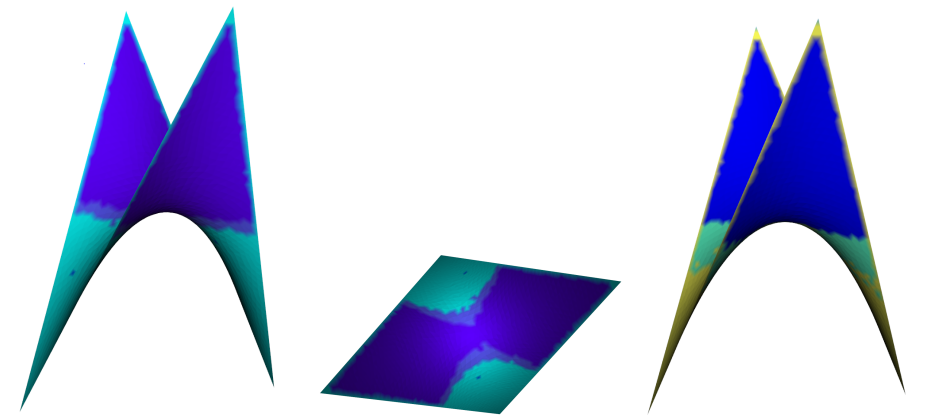
Figure 3-8: The logistic curve taken from Wikipedia [2008].

pact on vertex classification when applying a sigmoid function to our raw feature values. We will use the logistic function $s(x) = \frac{1}{1+e^{-kx}}$ [Weisstein, 2012b] where k is a constant that will be set to 10.0, 1.0 (Figure 3-8) or 0.1. The median value of each of the feature sets is used to re-center the feature so the transformed median value was 0 and then applied $s(x)$ to the raw values before feeding them into the SOM as training data or feature vectors to be classified. The median was chosen because extreme values in the feature data set would skew the mean.

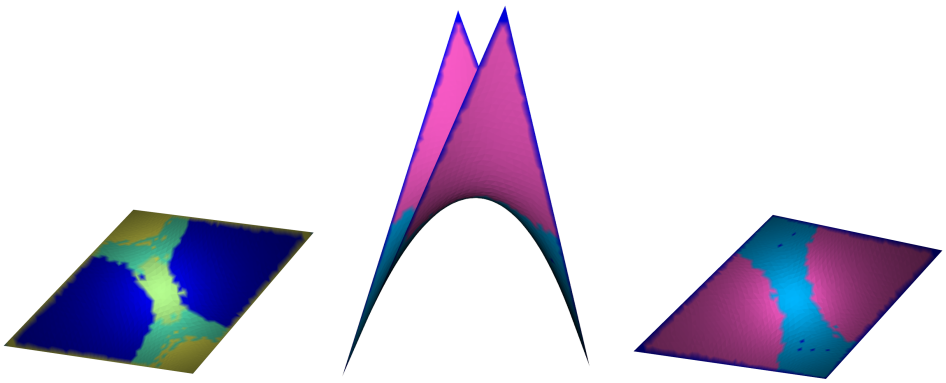
We trained the SOM using the set of exemplar surfaces shown in Figure 3-1 on page 56 using a 4×4 SOM trained with the features $HKSC$ with 10 coarse and 0 fine training epochs. These were the same parameters used in Section 3.1 on Page 58.

The results of the SOM trained with data scaled using the sigmoid function in Figure 3-9(c) show more details compared to those obtained with the raw max-min initialization shown in Figures 3-5(c) and 3-5(d). The results for $k = 0.1$ and $k = 10.0$ do not compare as favourably as they both lack detail in the saddle region at the centre of the mesh and in the case of $k = 0.1$ can only tell the difference between concave and convex regions.

We compared the results with the other exemplar surfaces trained with min-max scaling shown in Figure 3-10. The results highlighted the problem of determining both an appropriate k value and the appropriate function used to center the values around the

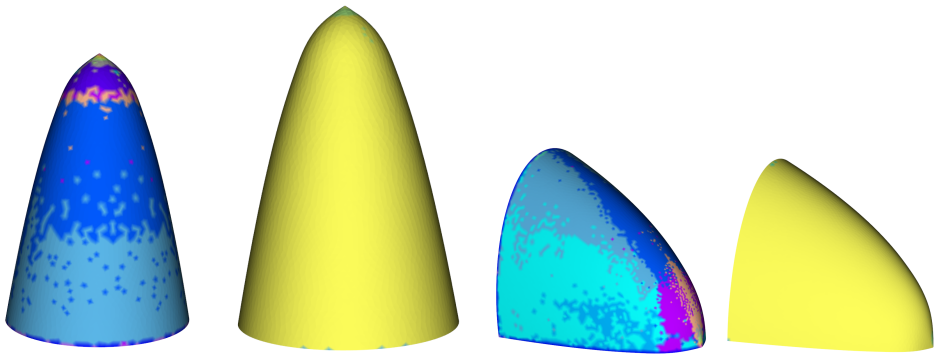


(a) $k = 10.0$ (side) (b) $k = 10.0$ (top) (c) $k = 1.0$ (side)



(d) $k = 1.0$ (top) (e) $k = 0.1$ (side) (f) $k = 0.1$ (top)

Figure 3-9: Results of classifying the hyperbolic paraobloid with different values of k for the sigmoid function.



(a) Min-Max (b) Sigmoid $k = 1.0$ (c) Min-Max (d) Sigmoid $k = 1.0$

Figure 3-10: Comparison of results of classifying the vertices of a paraboloid and elliptic paraboloid between min-max and sigmoid scaling with $k = 1.0$.

midpoint of the function.

For some surface types the curvature metrics measured on the surface may be clustered around multiple points making the decision of what values to transform difficult. Metrics such as S would be ideal for this pre-processing technique as they have a well defined range, but for other values such as k_1, k_2, H and K the values have a broad range and could be clustered at any point in the set of values. Rather than add more parameters to our experiments by using the $s(x)$ function with different parameter sets per feature we will continue to use the simple min-max scaling function throughout our experiments and allow the SOM to identify the clusters as this has already been shown to provide reasonable results.

3.1.3 Feature selection

The previous Section focussed on the behaviour of the SOM for a fixed feature set. We will now introduce different feature combinations to observe at which point the Q error stabilises for a given feature set and SOM size. The behaviour of the SOM will be different for each set of feature combinations as each feature has its own distinct numerical prop-

erties. For example the Gaussian curvature metric K is calculated as the product of the principal curvatures, whereas the shape index is a projection of the principal curvatures k_1 and k_2 into a polar co-ordinate space.

Using the results of the previous Section for the feature set $K\ H\ C$ and S we will reduce the range of SOM sizes to be 2×2 through to 20×20 as the Q error did not show significant change in the higher SOM sizes for these features. The feature combinations we will use in this section are:

- KC
- KH
- KHC
- KHk_1
- KHk_1k_2
- KHk_2
- KHS
- $KHSC$
- $KHSCk_1k_2$
- Kk_1
- Kk_2
- KS
- k_1k_2
- k_1S
- k_2S
- SC

- SCk_1k_2

The results of training a SOM with varying sizes and feature sets is illustrated in Figure 3-11 and shows that regardless of which curvature metric based feature set we used the Q error converges to a low value after a modest increase in the size of the SOM. The feature sets that converge at the fastest rate have a small number of features such as $H\ K$ and $K\ k_1$. The feature sets with a larger number of features such as $H\ K\ S\ C\ k_1\ k_2$ and $S\ C\ k_1\ k_2$ are the slowest to converge to a low Q error as the SOM size increases. This behaviour is expected as the SOM is having to group features in more dimensions as the feature count increases.

To better understand the behaviour of Q for all the feature sets as the SOM size increases, we examined the mean Q error for all of the feature sets and found that Q approached 0 as the SOM size increased (Figure 3-11). The convergent behaviour makes sense as at some point there will be a neuron for each feature combination in the training data set. Since the mean Euclidean feature distance is already near 0.01 at 7×7 there is little to gain when reducing the Q error by increasing the SOM size. Minors improvement in Q error comes at a significant computational cost as SOMs require O(xy) Euclidean feature distance calculations to find the BMU of each input feature in a SOM of xy neurons.

As in the previous Section we also examined the number of neurons fired for each of the SOMs when classifying the training data set's curvature based metrics for each of the vertices. The range of values for each of the feature and SOM size combinations are plotted in Figure 3-12. We found that for some feature sets the percentage of neurons fired was consistently above 90 percent whereas other feature combinations rarely passed the 90 percent mark. The feature combinations that consistently fired the higher rate of neurons during training, were all pairs apart from one set of four features and the metrics k_1, k_2, C, H and S all featured. The feature sets that didn't manage to fire such a high number of neurons all contained G. To visualise this difference we plotted the U-matrix for the feature set that produced one of the highest rates of neurons fired during training ($H\ S$) and when the feature set that produced the lowest rate ($K\ H$) for the SOM size

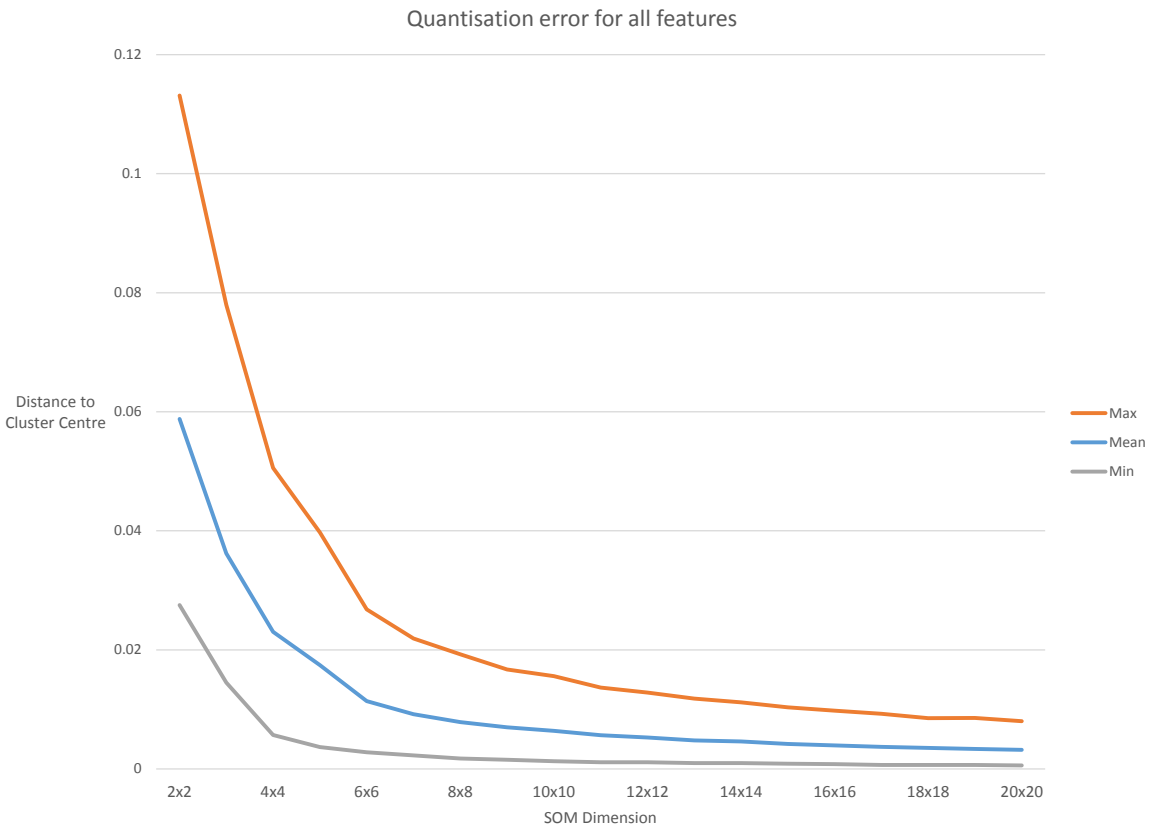


Figure 3-11: Mean Quantisation error aggregated across all feature sets

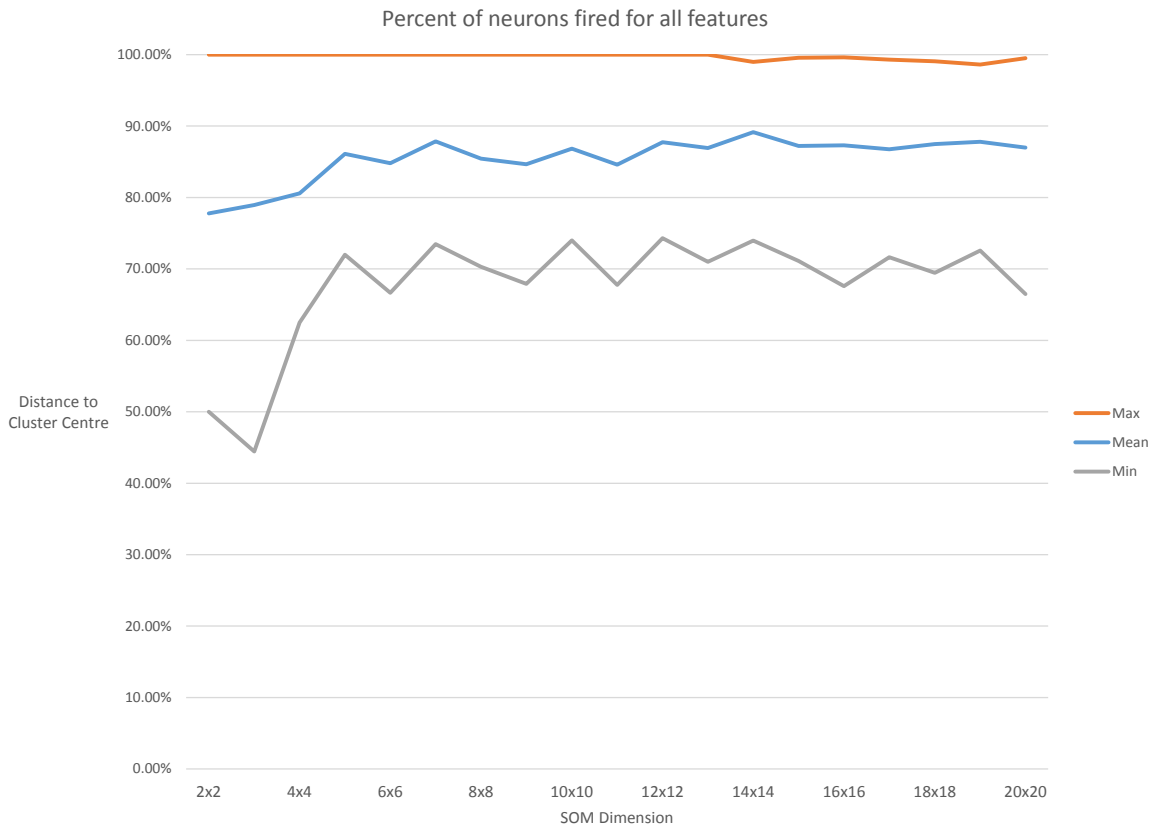
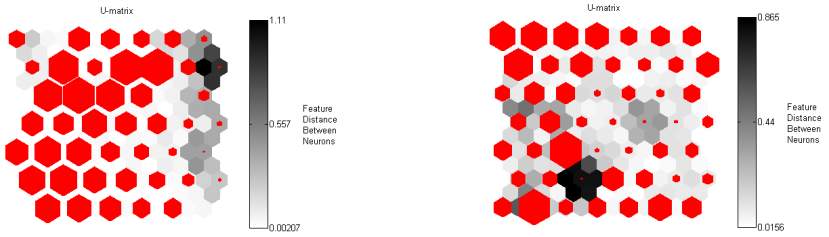


Figure 3-12: Percentage of neurons fired during classification



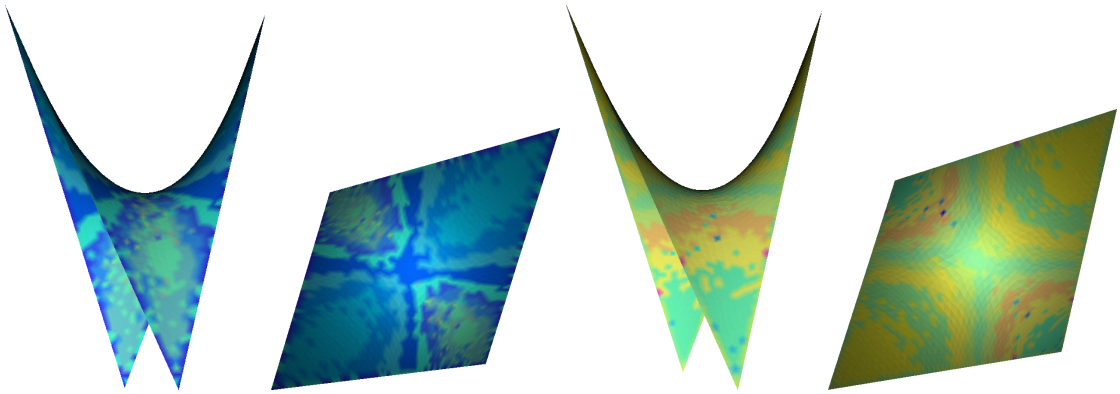
(a) $K\ H$ (b) $H\ S$

Figure 3-13: BMU hit distribution for the exemplar training data set for 7×7 SOMs

7×7 in Figure 3-13.

Upon examination, the U-matrices for the two feature sets differ in two ways. The first we have already discussed; the $K\ H$ SOM doesn't fire all neurons during classification. The second is that the $H\ S$ SOM has a greater number of neurons that have values sufficiently different to their neighbours that the space between them is marked in grey or black, whereas the $K\ H$ SOM has large regions of similar neurons with the majority of neurons having a non-white feature distance metric along the edges. K is less stable than H because it is the product of k_1 and k_2 so it is more likely to be affected by variations in these values compared to H which is the mean of the two. To understand how small feature distances between neurons affects how the SOM classifies vertices we display images of the hyperbolic paraboloid, used in the training data set (Figure 3-1) with the vertices coloured by their BMU.

The projection of the BMU color mapping onto the hyperbolic paraboloid for the two 7×7 SOMs in Figure 3-14 illustrates how noise is introduced when the SOM consists of neurons that are close to each other in feature space. The $K\ H$ SOM has a smaller feature distance between each of the neurons which translates into a SOM with a diminished ability to discern between difference curvature types. The regions visible in the side views of the hyperbolic paraboloid appear to be more clearly defined in the $H\ S$ model where there are more consistent bands of colour compared to the speckled blue sections of the $K\ H$ side view. The $H\ S$ feature set delivers relatively cleaner results because fea-



(a) K H Side (b) K H Top (c) H S Side (d) H S Side

Figure 3-14: View of Hyperbolic paraboloids with vertices coloured by BMU

ture S is independent of magnitude as discussed in Section 2.2 on page 28. The mean curvature H does introduce a magnitude component hence the banding effect on the surface. In contrast the combination of K and H results in two magnitude based descriptors that can result in different neurons firing for similar curvature types, K being especially susceptible to this problem since it is calculated as the product of k_1 and k_2 whereas H more stable as it is the mean of k_1 and k_2.

3.1.4 Training Iterations

The final parameter to consider is the number of coarse and fine training iterations to use. The coarse training phase has an effect on the neuron weights across the whole SOM whereas the fine training phase affects only the neurons in a small neighbourhood. To evaluate the effect of different training iteration parameters we used the feature set K H and varied both the coarse and fine training iterations as factors of the total number of neurons in the training data set. All combinations of 0, 1, 10 and 100 training epochs for both the fine and coarse training epoch parameters are evaluated. One training epoch consists of all training data from the set being fed through the SOM once.

We first examined the mean Q error when training the entire feature set with the SOM

with the results plotted in Figure 3-15. The Q error is at its highest when no coarse or fine iterations are used in the training process. This is expected as the SOM hasn't adapted itself to the training data. When there are no fine training iterations the Q error for each of the result sets was ranked by the number of coarse training iterations, 0 epochs producing the highest Q error and 100 training epochs producing the lowest.

When the number of fine training iterations was raised to be equal to the number of features within the training data set (1 epoch) each of the SOM's mean Q error values converged to a similar value, even for the SOM with no coarse training iterations. Once the number of fine iterations goes beyond 1 epoch the mean Q error across all of the coarse iteration values are similar as the fine training iteration value increases. The general trend of the Q error for each of these training combinations beyond 0 coarse iterations is that as the number of coarse and fine training iterations increases the Q error decreases initially and then stabilises. Interestingly the Q error for the maximum number of coarse training iterations (100 epochs) and 0 fine training epochs is comparable to the Q error for the results of the SOMs with the maximum fine training iterations, but without the additional computation required for the fine training iterations. The cost of performing a single fine training iteration is less than that of a coarse training iteration as the update function on average affects a smaller area for the fine iteration. This makes the use of only fine training iterations an attractive option.

To develop a deeper understating of how each SOM is behaving when classifying the training data set we plotted the percentage of neurons fired in each SOM during the classification process (Figure 3-16). As with the plot of Q error, the result of using a SOM that has not been trained at all was an outlier and in this case only 10 percent of the neurons were fired. For each of the sets of training iterations used the percentage of neurons fired increased with the number of fine training iterations with the exception of the 10 coarse training epoch set which remained relatively stable for all fine training iteration combinations.

To illustrate the difference between an untrained and trained SOM we projected the neu-

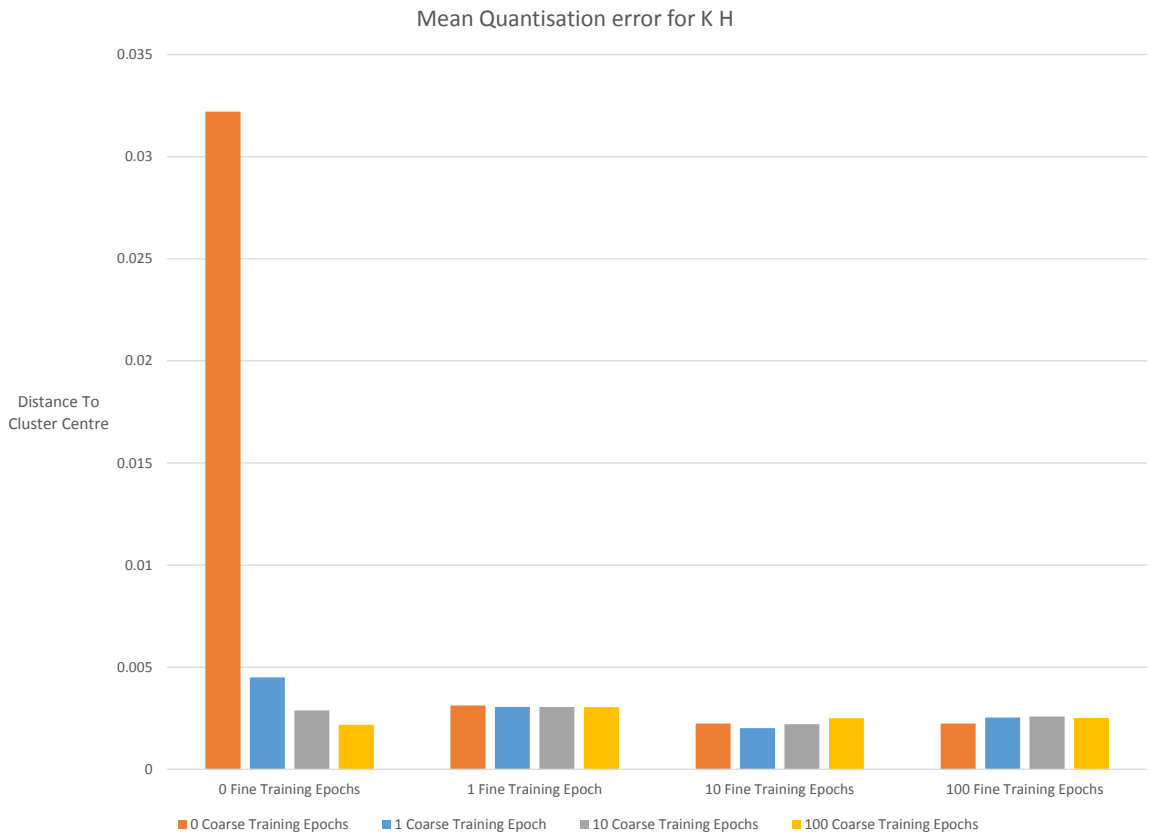


Figure 3-15: Mean Q error for training data set with two 7×7 SOMs trained with $K\ H$

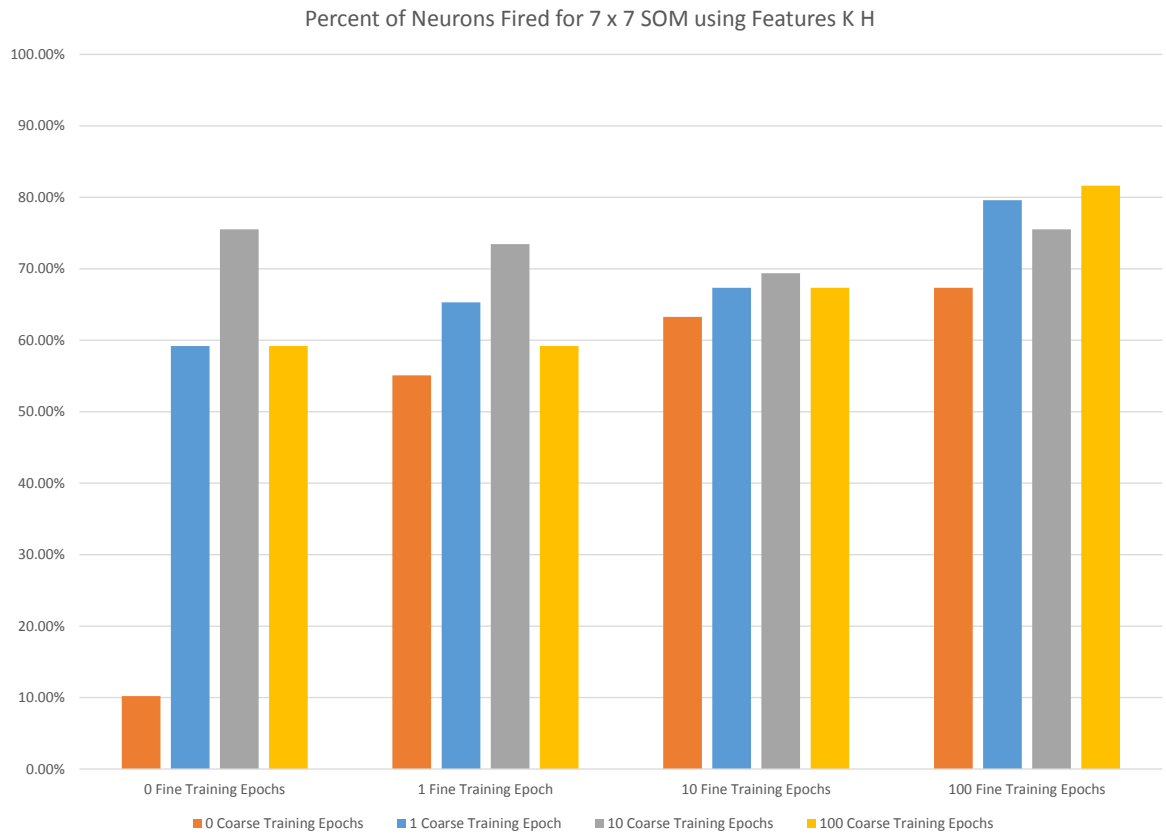
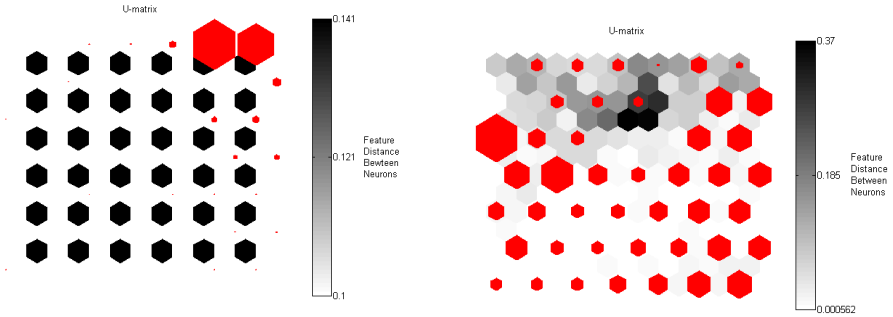


Figure 3-16: Percent of neurons fired for training data set with two 7×7 SOMs trained with $K\ H$

ral response generated during the training process onto the U-matrix for the untrained SOM and the SOM trained using 10 coarse and 1 fine training epochs as shown in Figure 3-17. For the untrained SOM the lack of stimuli is illustrated by the uniform distance between each neuron in the U-matrix and the tight clustering of the feature vectors in the training data set marked in red. In contrast the trained SOM U-matrix in the same Figure (3-17) shows a more uniform distribution of neuron hits (in red) and a non-uniform distribution of distance between each of the neurons illustrating the self-organizing properties of the SOM.

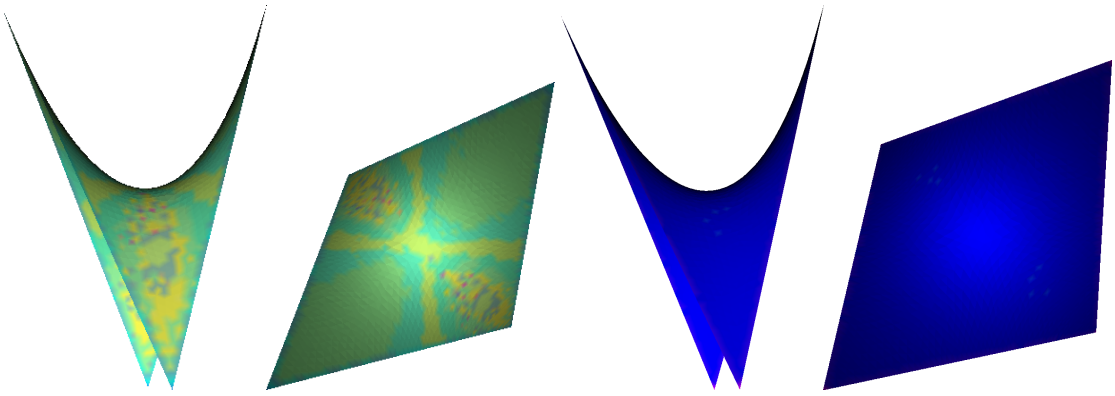


(a) 0 coarse and 0 fine training epochs (b) 10 coarse and 1 fine training epochs

Figure 3-17: BMU hit distribution for 7×7 SOMs using $K\ H$

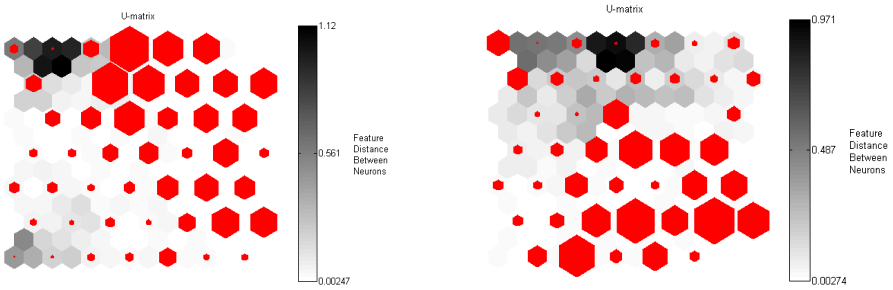
To explore the qualitative effect of the training process we rendered views of one of the exemplar surfaces, the hyperbolic paraboloid, with the vertices coloured by their BMU in Figure 3-18. The lack of hit distribution is clearly illustrated by the lack of colour variation in the surface coloured by the untrained SOM and the surface coloured by the trained SOM which marks distinct regions of varying curvature. We observed noisy regions in the concave sections which could be an artefact of the meshing, curvature calculation or training parameters.

Now that we have established the difference between the trained and untrained sets we focus on the difference between the results using two combinations of training iterations: 100 coarse epochs with 0 and 100 fine epochs respectively. Both combinations yielded a similar Q error but the SOM with the 100 fine epochs fired 80 percent of neurons during the training process compared to 60 percent for the SOM with 0 fine epochs.



(a) 10 coarse and 1 fine training epochs (b) 0 coarse and 0 fine training epochs

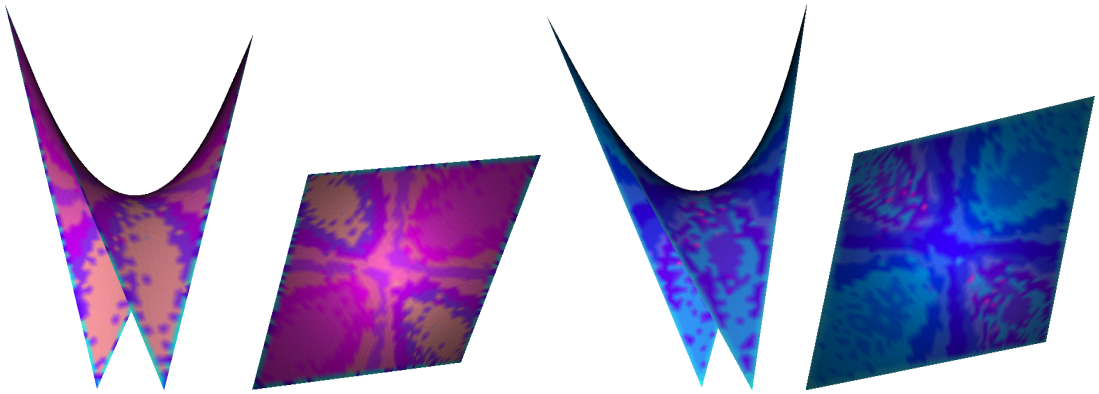
Figure 3-18: Hyperbolic Paraboloid with vertices coloured by BMU



(a) 100 coarse and 100 fine training epochs (b) 100 coarse and 0 fine training epochs

Figure 3-19: BMU hit distribution for 7×7 SOMs using $K\ H$

The U-matrix and neuron hit distribution for the two training iteration combinations are shown in Figure 3-19. Introducing the large number of fine training iterations increases the peak distance between the neurons and decreases the distance between similar neurons, represented by the decreased number of grey regions and increased number of white regions while the neurons surrounded by black (denoting a greater euclidean feature distance) remained relatively constant. With the introduction of the fine training iterations, neighbouring neurons that were similar, became even more similar and the distinct difference increased which makes sense given that the fine training process concentrates its effect on neuron weights in smaller locales compared to the coarse training phase that has an affect on a greater area.



(a) 100 coarse and 0 fine training epochs (b) 100 coarse and 100 fine training epochs

Figure 3-20: Hyperbolic Paraboloid with vertices coloured by BMU

To visualise the difference in classification behaviour when comparing the SOMs shown in Figure 3-19 we applied the BMU colouring scheme to the Hyperbolic Paraboloid, illustrated in Figure 3-20. Superficially the results are similar, but the SOM trained with 100 fine epochs shown in Figure 3-20(b) is over segmenting regions that appear as a single colour in Figure 3-20(a). For example, the purple/pink regions in the bottom-left and top right hand corners of the second image in 3-20(a) appear as one section, but the same region in 3-20(b) is broken down into more regions. The same holds for the top-left and bottom right sections of the same images. The increased fine training iteration count enables the SOM to be more sensitive the differences slightly different curvature characteristics upon inspection which is an indication that this SOM has been overfitted to the training data. As with the results shown in Figure 3-18 the noise in the concave sections persists.

3.2 Bimba Con Nastrino

The 3D model titled ”Bimba Con Nastrino” (Figure 3-21) from the AIM@SHAPE repository [Falcidieno et al., 2006] is used to demonstrate how the SOM can be used to identify different curvature types on a free-form surface. We selected this model because it rep-

Figure 3-21: Bimba Con Nastrino solid rendering. [Falcidieno, 2009]

resents many different kinds of regions varying from the broad sweep of the cheek bone, the complex self-similar regions within the braided hair and the symmetrical features of the face. The model contains 74764 vertices, 224286 edges and 149524 faces, was acquired using a laser scanner, processed to remove holes, smoothed and then re-meshed to make the face sizes and edges as uniform in length as possible thus avoiding some potential bias in the representation of the surface. Figure 3-22 shows the mesh representation for the whole model and for a small area. This model had already been pre-processed and converted into a uniformly meshed OOGL file so we imported the model and extracted its curvature properties using our own Java code.

3.2.1 SOM Size

The SOM was trained on the 3D model in Figure 3-21 using the feature set $KHSC$ because they did not appear to result in an overfitted SOM. We found that the percentage of SOM Neurons fired during the classification process plotted in Figure 3-23 remained high for all SOM sizes from 2×2 to 80×80. The downward trend shows that as the SOM increases in size less neurons are being fired, making the additional neurons redundant. In comparison the SOM trained on the exemplar surfaces in Figure 3-1 had a very sharp drop in the number of neurons fired for the 3D model as the SOM increased

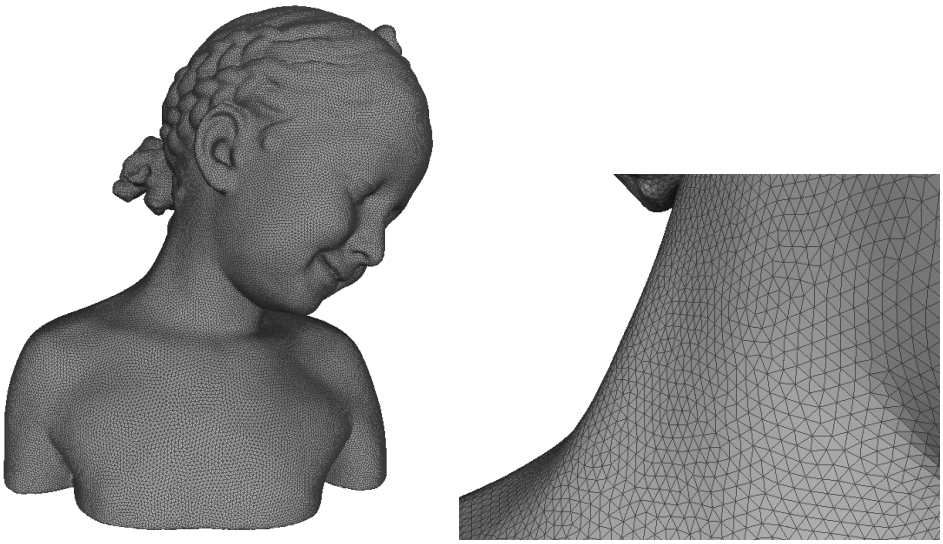


Figure 3-22: Bimba Con Nastrino mesh representation with detail. [Falcidieno, 2009]

in size (Figure 3-24) because the 3D model training data set better represents the models being segmented compared to the exemplar surfaces. To evaluate how well the neurons represented the vertices in each cluster we calculated the mean Quantisation error (Figure 3-25) and found it decreased rapidly as the SOM size increased. Given the search for a BMU in a SOM is bound by the number of neurons and the complexity of that search for a BMU of a $n \times n$ SOM is $\Theta(n^2)$ and as Q decreases the SOM is more likely to over-fit data we would not want to pursue zero quantisation error.

Using the exemplar data set illustrated in Figure 3-1 and the same training parameters we processed the 3D model and measured the Q error when classifying the vertices. We found that regardless of the SOM size the Q error remained constant and there was no noticeable trend upwards or downwards as the SOM increased in size (Figure 3-26.) This illustrates the difference between the feature distributions of the exemplar data set and the 3D model being classified.

Another way to evaluate how well a SOM represents the input data set is to visualize the distribution of neuron hits when that data set is classified. As the neuron weights are readjusted when stimulated by input data, the SOM will more closely match the input data

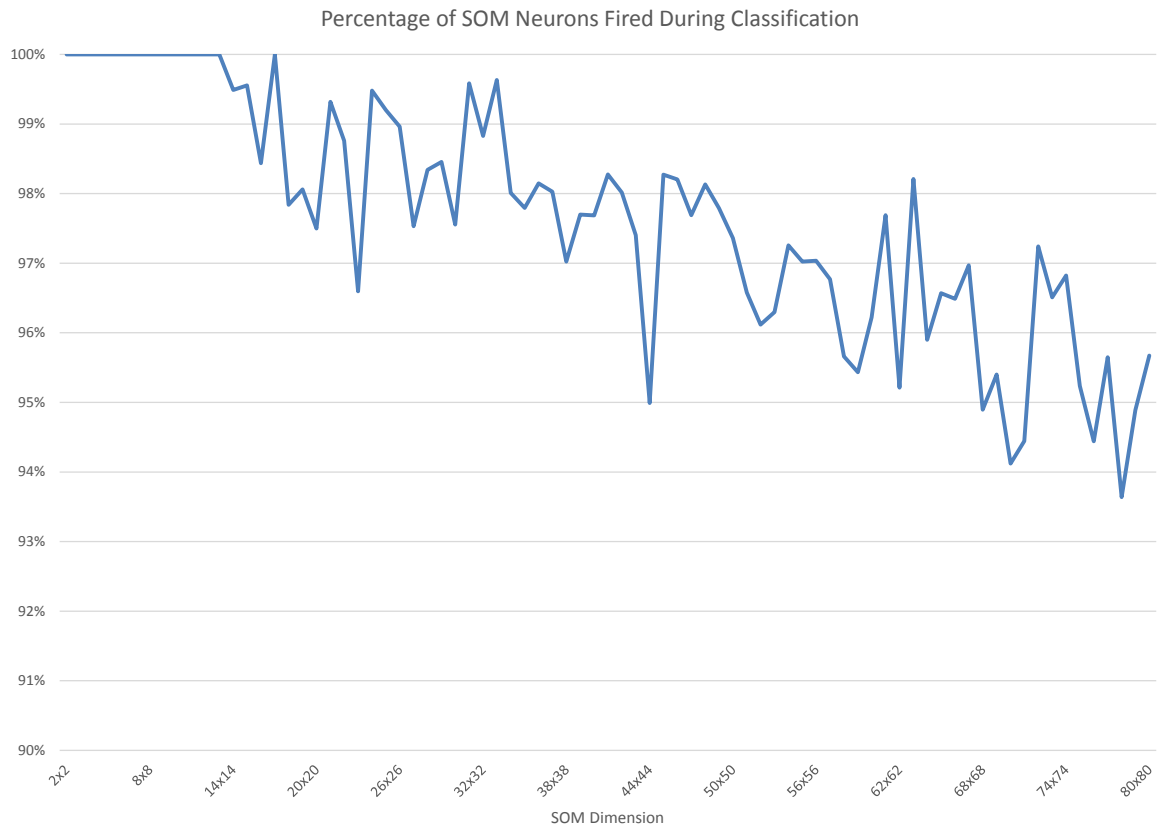


Figure 3-23: Percentage of SOM neurons fired for 3D model training set during classification of surfaces.

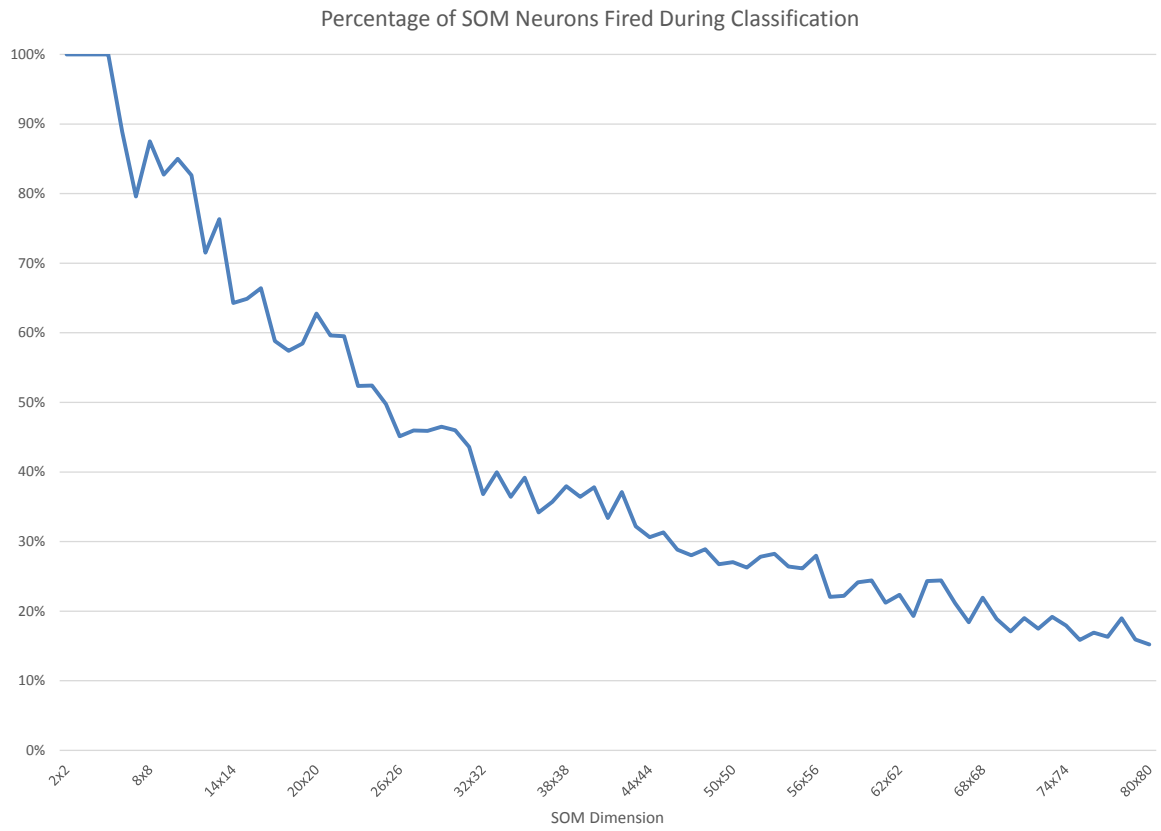


Figure 3-24: Percentage of SOM neurons fired for exemplar training set during classification of surfaces.

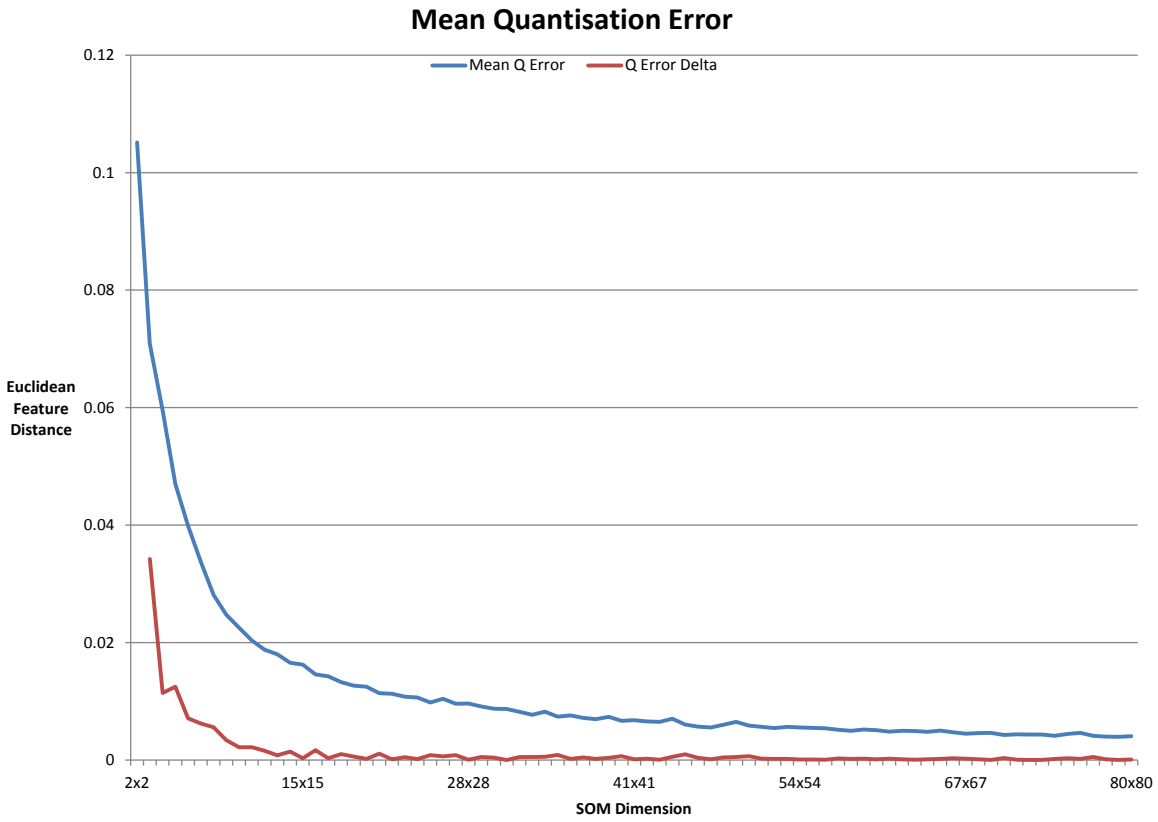


Figure 3-25: Plot of mean Quantisation error for 3D model training set.

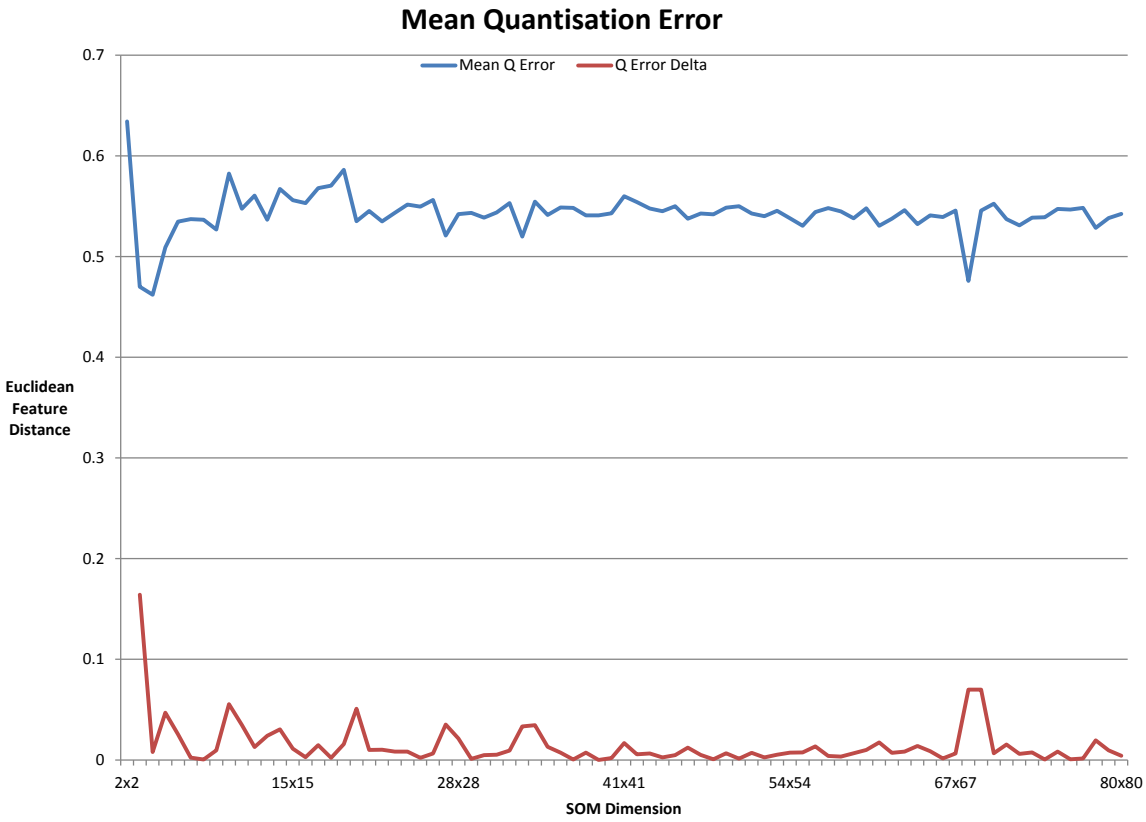


Figure 3-26: Plot of mean Quantisation error for exemplar training set.

set. Since the number of neurons in a SOM is fixed, the distribution of neurons being fired when classifying the same dataset used for training will become more evenly distributed as the number of neurons increases. After a given point the increase in SOM size will cause the distribution of neurons firing to change as the dataset is then represented by too many clusters.

The 3D models with their vertices coloured to show the mapping from colour to neuron and distribution of neuron hits projected onto the U-matrix for the increasing SOM size are illustrated in Figure 3-27. These figures show that regardless of the size of the SOM within the interval of 2×2 to 20×20 is able to cluster the input data with a reasonably even distribution.

To visualise the cluster membership of a vertex we will use a colour palette that maps the vertices cluster membership (BMU) to a colour. The mapping is arranged on a cartesian plane to mirror the 2D co-ordinate system used to name the SOM neurons with the upper left hand corner being the origin $(0, 0)$. Examples of this mapping technique are shown in Figure 3-3 on page 58.

To understand what surface type is represented by each neuron we consider the 2×2 neuron SOM in Figure 3-27(a) which is able to represent four different surface types.

- $(0, 0)$ (Dark Blue) Convex with curvature in both directions with one direction being more curved than the other: Edge of the ears, nose, lips and the larger knots of hair.
- $(1, 0)$ (Purple) Concave with curvature in one or both directions: Above the collar bone, neck, between each eye and nose and in the crevices of the braided hair.
- $(0, 1)$ (Light Blue) Convex with curvature both directions: Shoulders, cheeks, forehead, chest and chin.
- $(1, 1)$ (Grey) Convex with the curvature being in one direction: Nose and neck areas.

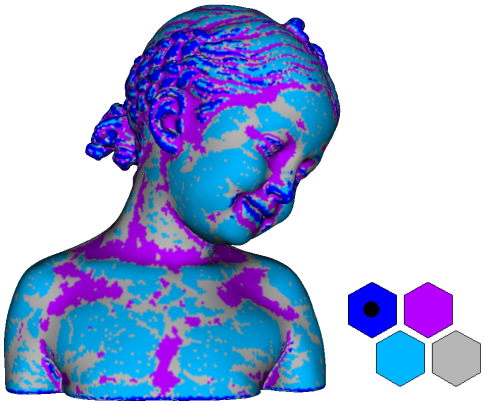
A limited surface type vocabulary of four neurons leads to vertices that have different

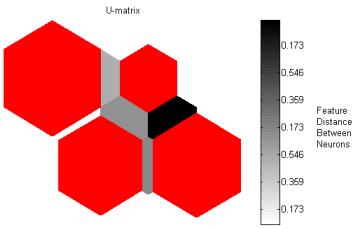
curvature properties being classified as the same type. Increasing the number of neurons yields better results. An example of this is illustrated by the vertices about tip of the nose using the 2×2 SOM being considered the same type as the gently curved cheek but when the number of neurons is increased they are not the same as illustrated in Figures 3-27(b) and 3-27(c).

The relatively poor results generated when training the SOM with a set of exemplars (Figure 3-28) produced regions that were poorly defined in comparison to any of the results when the SOM is trained on the 3D model being classified in Figure 3-1. The majority of the neurons being fired when the 3D model is being classified exist in the same region of the SOM. This illustrates the lack of feature diversity described by the SOM when different regions, that to the naked eye, are dissimilar are considered the same.

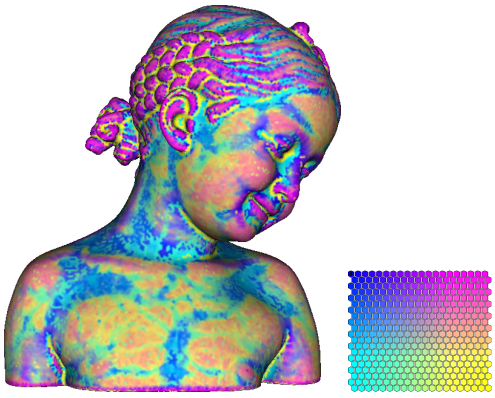
As the number of neurons increases the variations in curvature that can be described by the SOM increases. Regions become difficult to differentiate using simple shapes and we become increasingly reliant on the colours used to represent the cluster membership of each vertex. As the quantisation error begins to flatten out with the SOM increasing in size, as shown in Figure 3-25, the increased computation required to train these large SOMs gives us smoother transitions between regions of different curvature by increasing the number of vertex types that can be represented by the SOM. This can be seen when examining the neck region above the collarbone of the model and noting the transition from a largely single colour representation in Figure 3-27(a) through to the smoother colour transitions in Figure 3-27(c).

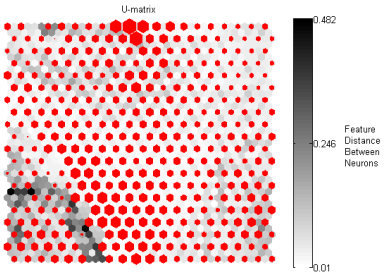
The complexity of all algorithms that use the SOM is $\Theta(n^2)$. Given the computational expense of increasing the size of the SOM and our results, it is not worth using a SOM larger than 16×16 as there is a minor decrease in Q error (Figure 3-25) and a greater likelihood of overfitting data to the SOM.



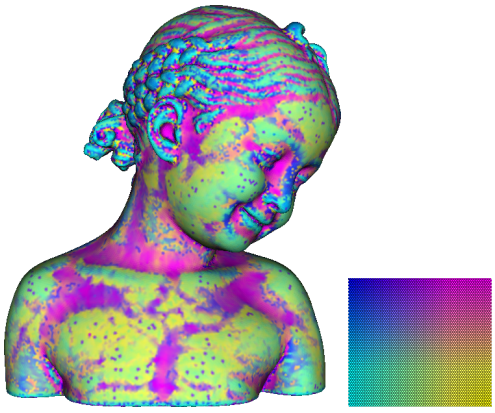


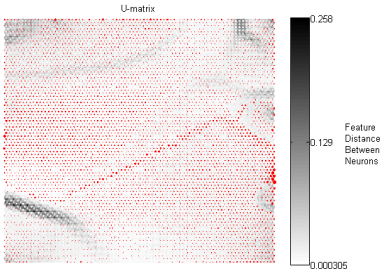
(a) 2×2





(b) 20×20





(c) 80×80

Figure 3-27: Bimba Con Nastrino - Classified with SOM trained on the model itself using the features $KHSC$, showing classifier colour palette and projection of neuron hit count onto the U-matrix

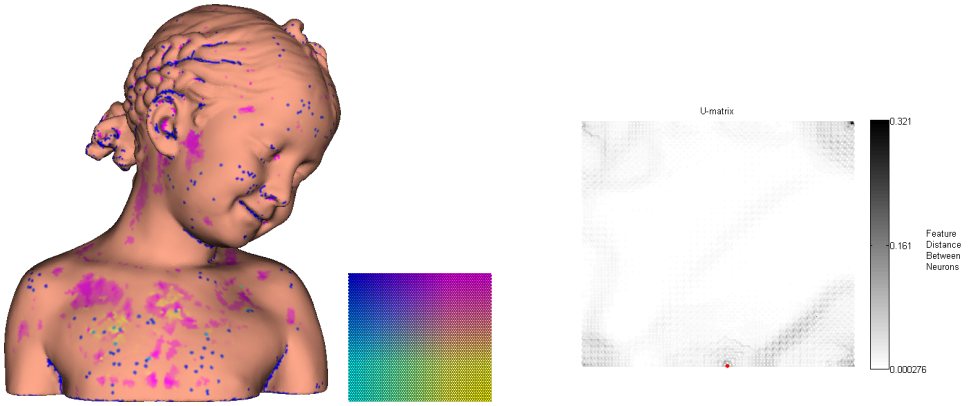


Figure 3-28: Bimba Con Nastrino - Classified with a 80×80 SOM trained on the exemplars in Figure 3-1 itself, showing classifier colour palette and projection of neuron hit count onto the U-matrix

3.2.2 Feature Selection

The numerical stability and quality of features has a strong influence on how well the SOM can differentiate between regions of different curvature. Curvature metrics that are susceptible to noise or are numerically unstable degrade the ability of the SOM to cluster effectively.

The features used in these experiments vary from the relatively raw metrics of the two principal curvatures k_1 and k_2, to the Gaussian K and mean curvature H which can be derived from the principal curvatures, and the curvedness C and shape index S, which are projections of the principal curvatures into different co-ordinate spaces. These have been used in MacLennan et al. [2006], MacLennan and West [2008] and Koh et al. [1993] as features to classify vertices using self-organizing maps.

To evaluate the effectiveness of the curvature metrics in different combinations as features used to train the SOM and classify vertices we used a 16×16 SOM which worked well for the feature combination of K H C and S used in Section 3.2.1. We found that there were three distinctive groups of feature sets.

The first group represents the results with the least quantisation error and consists solely of pairs of features with Gaussian curvature present in each of these pairs. The second group contains a mix of the features and the third contains larger sets of features that all contain the Shape Index and Curvedness metrics. We used this set of combinations to constrain ourselves to a reasonable number of experimental parameters.

To illustrate the difference between these sets of results we projected the vertex cluster memberships for each vertex onto the 3D model using the colour mapping illustrated in Figure 3-29 and plotted the distribution of neurons fired onto the U-matrix of the SOM for the given set of features in Figures 3-30, 3-31 and 3-32.

The example taken from the set of low quantisation error feature sets shows many regions where the transition between vertex types is abrupt as illustrated by the speckled colour scheme across the whole 3D model in Figure 3-30. What could be considered visually subtle differences between the curvature characteristics of adjacent vertices are significant according to the classification results due to the sensitivity of the SOM and its ability to accurately cluster the curvature features K and C. Excessive sensitivity of the SOM is an indication that the data has been over-fitted. The results for the feature set K was far worse with a large number of very small regions. This feature by itself is not a good choice when classifying vertices but may be acceptable when combined with other feature. Based upon these results the use of a single feature will result in over-fitting of data and if the feature K is present in a set of two features there is a high probablility of the SOM being over-fitted. When the number of features is increased the SOM has to map more dimensions into 2D space resulting in a higher quantisation error. This is reflected in our results where the quantization error for sets of five features are all in the upper half of all the results and the results for the complete set of six features has the greatest quantisation error.

Results for the feature set belonging to the group of features that give a medium level of Q error illustrated in Figure 3-31 shows smoother transitions between regions of similar curvature types. Comparing the chest and collarbone region of the results for this feature

set and the model in Figure 3-30 we see that the transitions between regions of different curvature characteristics are more distinct and the regions themselves appear more closely related. This close relationship is illustrated by the green region surrounded by yellow that marks the collarbone in Figure 3-31 in comparison to the speckled colour scheme in Figure 3-30. This distinct colouring marking cluster membership is taken even further in Figure 3-32 where the average quantisation error is very high. The collarbone section begins to lose detail, the region where collarbone lines up with the shoulder is shorter, and the lower central chest area which is coloured in a shade of blue contains less colour changes compared to the same region in 3-31. Colour represents different neurons being fired so less colour means fewer neurons of different types being fired.

We can identify clusters of features for these results by examining both the hit distribution (red) and intra-neural distance (shades of grey) in the U-matrices in Figures 3-30, 3-31 and 3-32. The feature set KC (Figure 3-30) produces a large cluster of only a few neurons that represents concave to flat regions represented by the large cluster of neuron hits at the bottom of the matrix with many smaller surrounding that cluster and a boundary bisecting the U-matrix across the top left hand corner which describes convex sections. This noisy result with many small clusters illustrates what happens when the SOM is over-fitted to the data set. Although this could be useful in some circumstances it is more desirable to have clusters that represent larger regions. As Q increases we see more clusters begin to develop in the U-matrix with less single neurons dominating the proportion of hits and greater separation between regions with high inter-neural distance and a cleaner differentiation between regions on the model in Figure 3-31. Further refinement in Figure 3-32, which produced one of the highest Q mean error values, shows a greater spread of hits across the neurons and for the first time multiple clusters of neurons with a higher proportion of hits as illustrated by the large number of hits in the top right hand corner and the lower-central region on the matrix. This spread of clusters is illustrated by the distinct colouring used to define each of the regions on the model.

When determining what the top 5 neurons were being fired by the input data we found that the feature sets with low and medium quantisation error required a large number

Figure 3-29: 16×16 Colour palette

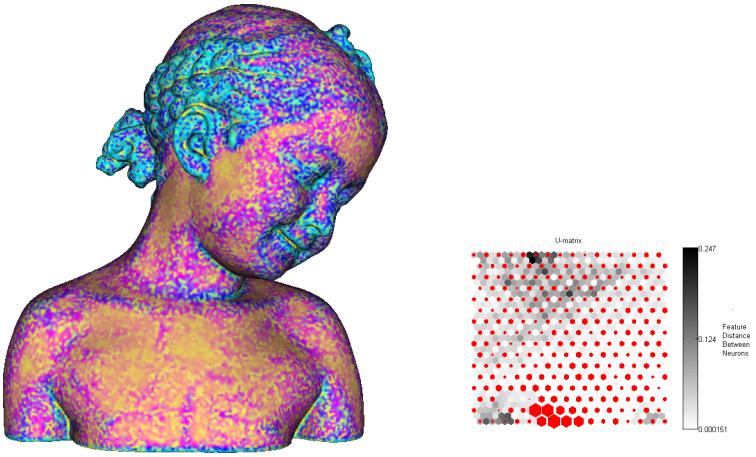


Figure 3-30: KC - sample from group with smallest quantisation error trained on Bimba Con Nastrino

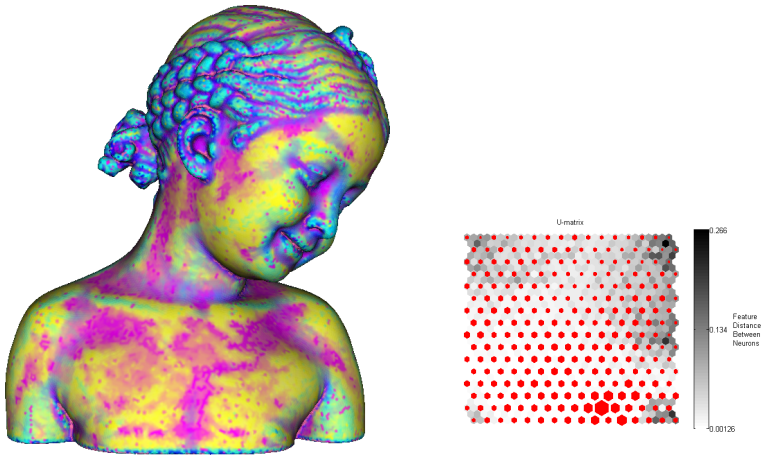


Figure 3-31: k_1k_2 - sample from group with average quantisation error trained on Bimba Con Nastrino

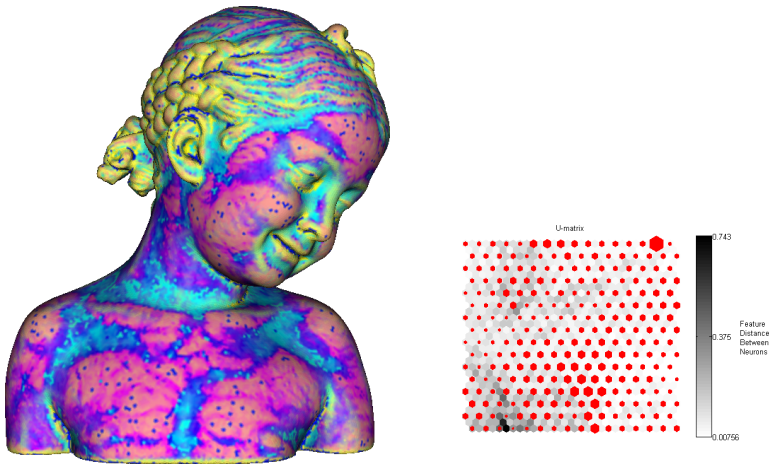


Figure 3-32: SCk_1k_2 - sample from group with largest quantisation error trained on Bimba Con Nastrino

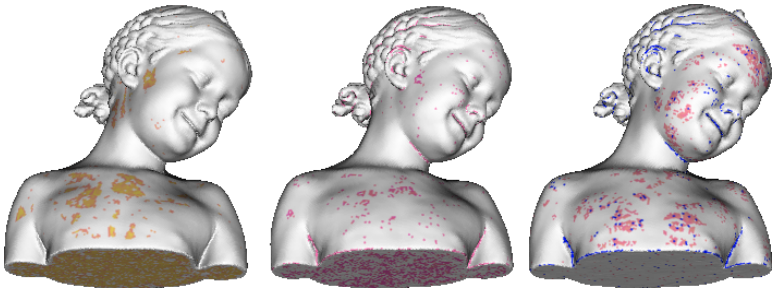


Figure 3-33: Top 5 neurons for the three feature sets KC, k_1k_2 and SCk_1k_2

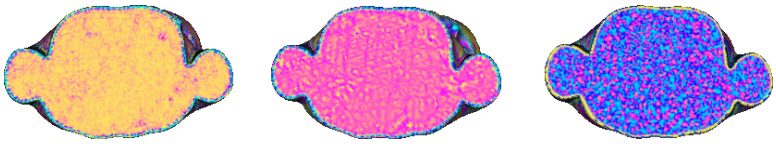


Figure 3-34: The underside of Bimba Con Nastrino for the 3 feature sets KC, k_1k_2 and SCk_1k_2

of neurons to represent the flat surface underneath the model and the high quantisation error feature set required less. The images of the top 5 neurons fired being projected on to the model are shown in Figure 3-33.

Features sets such as those used in the low and medium quantisation error examples have the desirable attribute of requiring less neurons to represent what was in effect a flat surface, meaning more clusters are available for representing regions containing different curvature properties that are more interesting than a flat surface. The feature set used in the high quantisation error set of results dedicated more neurons to represent flat areas, a sign that the SOM is overfitting data. Figure 3-34 shows the underside of the model coloured by vertex classification and when comparing this image with those for the top 5 neurons being fired shown in 3-33 you can see the larger number of neurons, that are distant from each other on the U-matrix (in Figures 3-30, 3-31 and 3-32), required to represent a flat surface.

The feature sets for the low and medium quantisation error results are able to represent the flat surface that makes up the base of the 3D model with the features K, C, k_1 and k_2.

The high quantisation error set combines the same metrics except K is substituted for S, the shape index. The shape index metric by definition has no value for planar surfaces which explains why this metric functions poorly when processing a surface with planar regions.

When examining the plot of mean quantisation error in Figure 3-35 find most combinations are single or pairs of features. The upper half of this set contains instances of C and S and the lower half is dominated by the presence of K. The different Q results indicate both how strongly different features correlate with each other and the abundance or lack of neurons to accommodate the different combinations within the set of values being classified by the SOM.

Since S and C are a representation of curvature in two components, the type and magnitude, they won't exhibit a strong correlation as different curvature types can share the same magnitude and conversely different types can share the same magnitude. The lower half contains feature combinations that are strongly correlated, H is the mean of k_1 and k_2 and K is the product of k_1 and k_2 so it is natural that these combinations would result in a low Q error.

When the number of features in the set increases the mean quantisation error increases. This is both a function of the increase in feature space and how different curvature metrics relate to each other. This is illustrated by the 12 feature sets that had the lowest mean Q error. They contained 6 instances of feature pairs and 6 instances of single features.

We found that for this set of experimental parameters the best results came from using pairs of curvature metrics, some which had been already used by Besl and Jain [1988] in a different way, and others which had not been combined in this way before. The feature sets used by Besl and Jain [1988], K and H (Gaussian and Mean curvature) and k_1 and k_2, performed well and were ranked 10^{th} and 23^{rd} respectively. It is important to note that we are using the feature pairs in a different way. Besl and Jain [1988] used sign based rules whereas we use the clusters of the SOM to describe the vertices which gives us a

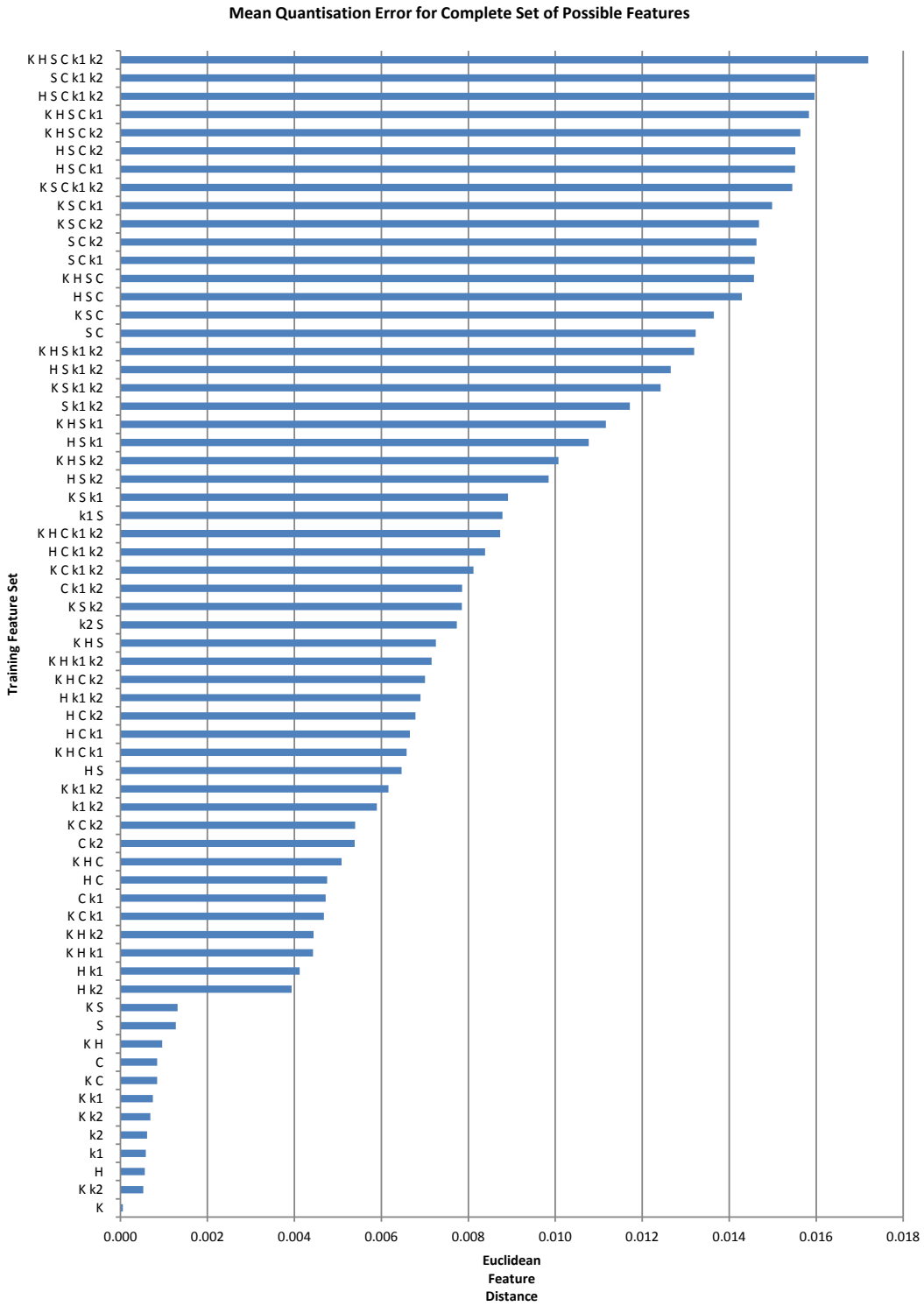


Figure 3-35: Mean Quantisation Error for 16×16 SOM with different features.

greater vocabulary than the 9 surface types defined in Tables 2.1 and 2.2 on pages 25 and 24.

When considering how well a SOM represents the input features and how this relates to the effectiveness of our classification based on this clustering technique we need to find a balance in the Q error that isn't so low that it over fits the data but doesn't under fit it. From this set of results it appears that using a mid-sized set of features that aren't strongly correlated, results in a SOM that strikes a balance between under and overfitting the data. Throughout Chapters 4 (Sections and 5 will demonstrate how Q impacts the results.

3.2.3 Training Iterations Parameters

Using the SOM size 16×16 and feature set of S and C we calculated the mean quantisation error from SOMs trained with different numbers of coarse and fine training iterations. These features were used as they resulted in a relatively high Q compared to other feature combinations leaving room for measurable improvement when the iteration parameters are varied. The coarse training process affects more of the SOM whereas the fine training process only affects a small number of neurons around the BMU for the current set of data being used to train the SOM (Page 15.)

We used multiples of the number of vertices, described as epochs, in the 3D model being used to train and visualise the SOM. The two parameters defining the number of fine and coarse training iterations had their values adjusted independently in order for us to determine the effect of different combinations.

In Figure 3-36 the results show that the parameter that has the greatest effect in isolation on reducing the mean quantisation error is the fine training iteration count, but the lowest Q error was attained when both iteration counts were at the largest values for this experiment. A large number of fine training epochs regardless of the number of coarse training epochs would result in a low mean quantisation error and a small number of

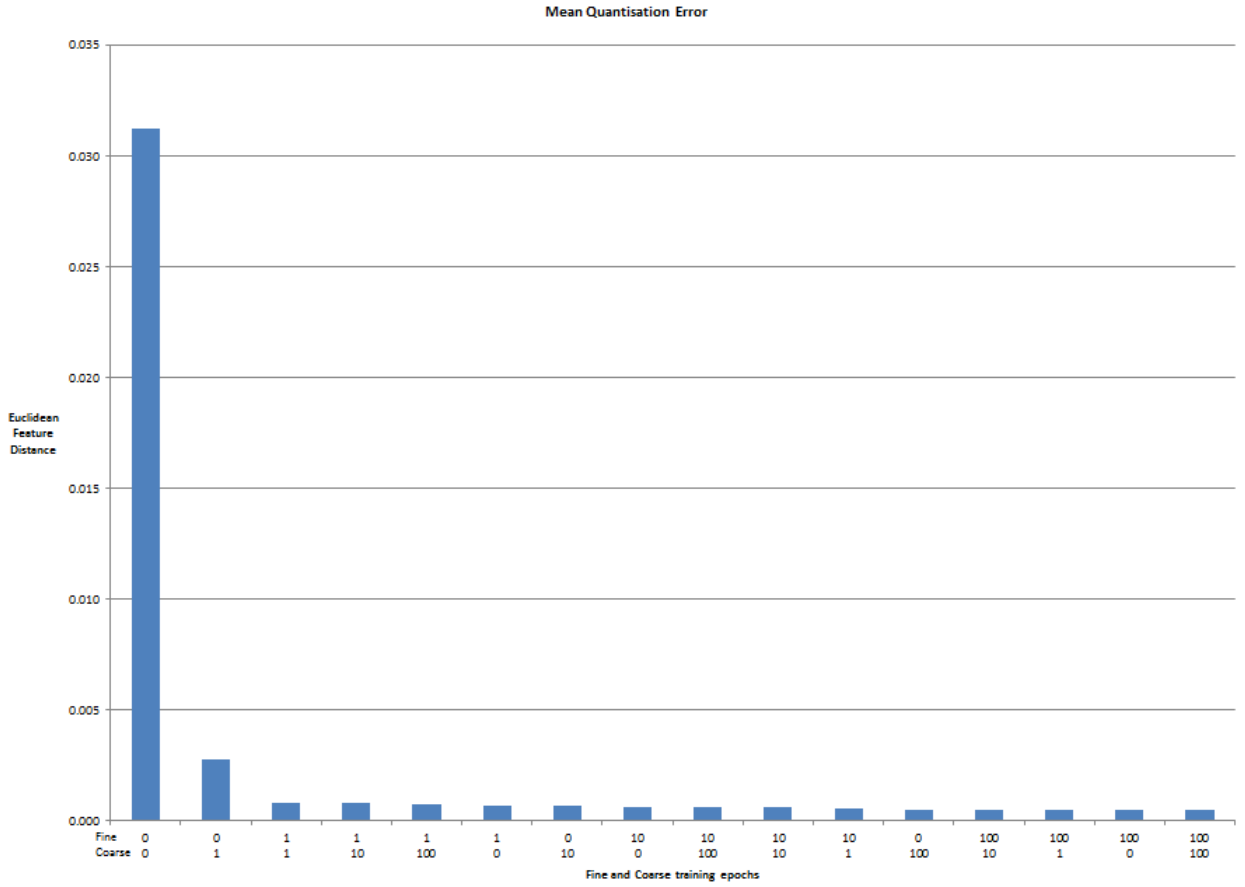


Figure 3-36: Mean quantisation error for the features S and C using a 16×16 SOM and different fine and coarse training iterations

coarse training epochs with little or no fine training epochs would result in a higher mean quantisation error.

When classifying the vertices we found that when considering the number of coarse and fine training iterations to use it is an all or nothing proposition when aiming for a low Q error. We can use the maximum number of feasible coarse and fine training iterations or only the maximum number of fine training iterations, the latter being more attractive since it requires far less computation. There wasn't much difference between the Q error across the results (excluding no training) so we can conclude that for this set of training parameters it doesn't take many training iterations for the SOM to converge on a stable solution for either training phase.

To see if our SOM can generalize to classify vertices on other models we applied the SOM used to generate the results in Figure 3-32 to other 3D models. The results of classifying the vertices on the Fu-Lion (Figure 3-37(a)) and Neptune (Figure 3-37(b)) sculptures are similar with respect to the types of surfaces recognized, how they are coloured with the lack of regions marked in yellow representing the braids of hair in Figure 3-32.

The Sappho bust shown in Figure 3-37(c) has a different appearance with the surface dominated by neurons represented by the hues of green and yellow with none of the neurons represented as blue through to purple that dominated the Neptune and Fu-Lion models. The shaft of Neptune's trident, which has a consistent curvature in one direction and zero curvature in the other direction is coloured purple and the base of the Sappho bust that shares similar properties but is coloured yellow and shows no neurons firing that are shared in common with Neptune.

The variation in the SOM response to each of these surfaces is explained by the presence of three measures that are sensitive to scale: k_1k_2 and C which is the root mean square of k_1k_2. The plots in Figure 3-39 illustrates how the mean value for scale sensitive feature C, a measure of curvature magnitude, has a broad range of values across the surfaces compared to the smaller range of mean values for S. The best result in our set of test

(a) Fu-Lion [Falcidieno et al., 2006] (b) Neptune [Falcidieno et al., 2006] (c) Sappho [Artec Group inc., 2012]

Figure 3-37: Models classified using SOM trained with the features SCk_1k_2, trained on Bimba Con Nastrino.

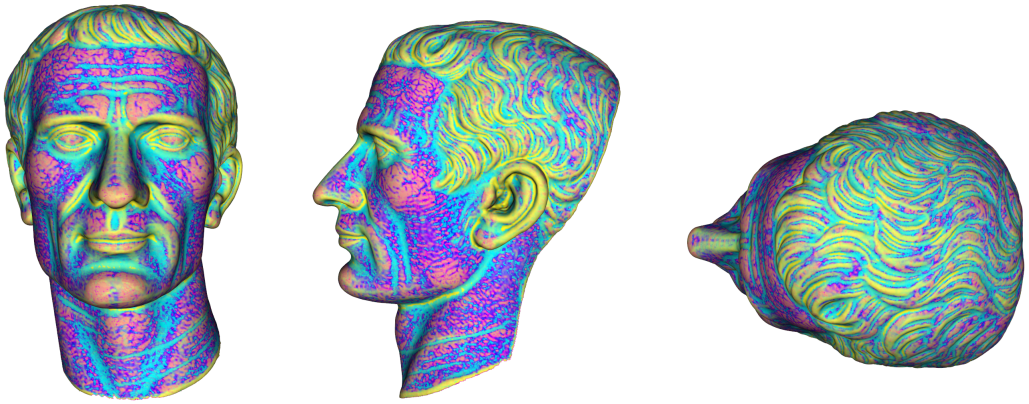
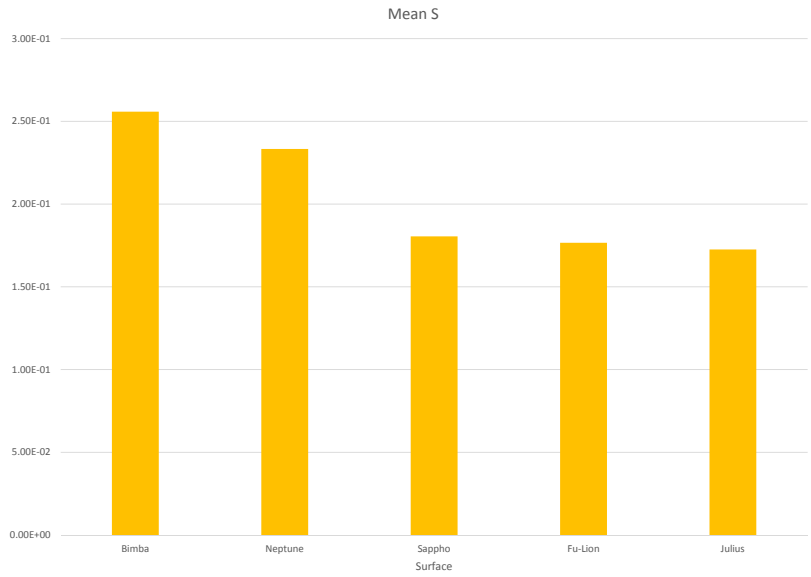
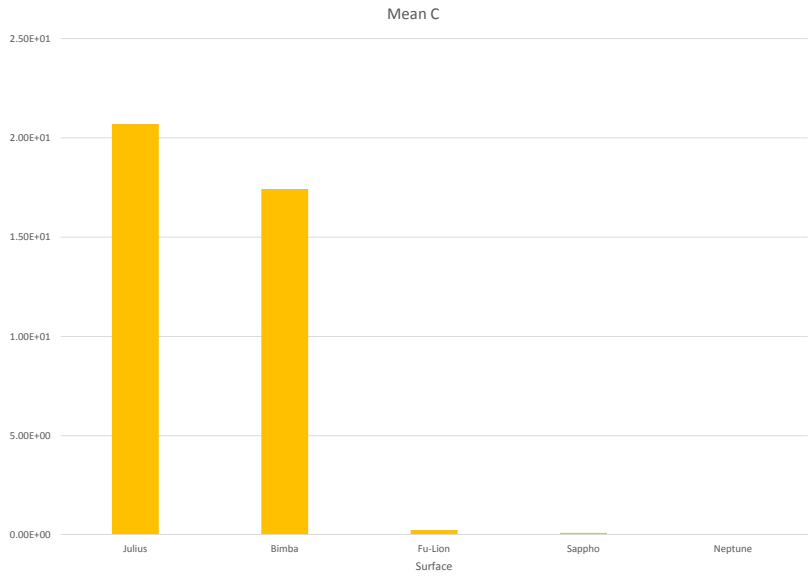


Figure 3-38: Julius classified using SOM trained with the features SCk_1k_2 [Falcidieno et al., 2006].

surfaces classified with the SOM trained on Bimba, shown in Figure 3-38, has a mean value for C similar to the Bimba model. The mean k_1 and k_2 values in Figure 3-40 also illustrates the differences between the curvature metrics from each of the surfaces. For both metrics the model in Figure 3-38 has similar k_1 and k_2 mean values compared to the



(a)



(b)

Figure 3-39: Mean feature values for the surfaces shown in Figures 3-32, 3-37 and 3-38

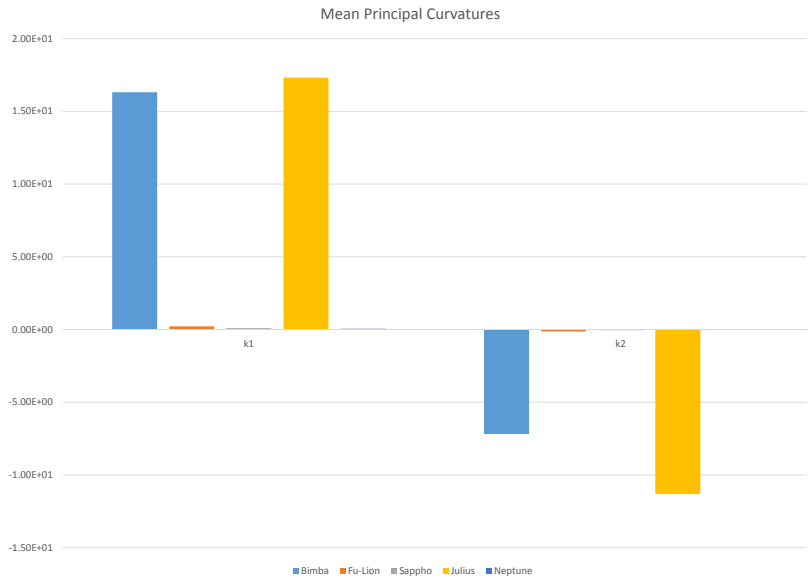


Figure 3-40: Mean feature values k_1 and k_2 from the surfaces shown in Figures 3-32, 3-37 and 3-38

training data set from the Bimba model.

3.3 Conclusion

We investigated the effect of SOM size, feature selection and training parameters when using a set of conic sections (paraboloids) by examining the quantitative measures of Q (using percentage error) of neurons fired along with qualitative measures using U-matrices representing the inter-neuron feature distance and neurons fired, along with images of the surfaces being classified with their vertices coloured by BMU. The paraboloids were used as a source of training data as they represent a wide variety of curvature types in a compact set of shapes.

We found that for the paraboloid training data set, as the SOM size increased the percentage of neurons fired decreased and the Q error stabilised. Regardless of the feature set chosen the Q error decayed at a similar rate but some feature combinations such as SCk_1k_2 and HS maintained a high percentage of neurons fired during the classification of training data whereas the feature sets KM and KS showed consistently low percent of neurons fired. When evaluating the sensitivity of the SOM to different coarse and fine training iterations we found that for most cases increasing the number of fine training iterations improved the Q error and percent of neurons fired. We also found that by increasing the fine training epoch count, the SOM's neural weights began to move closer for similar neurons, and neurons that were dissimilar moved even further apart but an excessive number of fine training epochs intoduced overfitting.

The SOMs generated from the training data set and a new set of SOMs trained on a 3D model were then used to classify the vertices of a scanned 3D model. Alternating between the two sets of training data, varying SOM sizes and feature sets and training iteration counts using quantisation error as quality metrics, we were able to illustrate the effect that these parameters have on the mean quantisation error. When using a SOM as a classifier with the aim of reducing quantisation error we found that a 16×16 SOM with the features S and C, the largest number of coarse and fine training iterations and the 3D model itself as training data we found gave the best results. When evaluating the effectiveness of the training data set the SOM trained on the 3D model produced better results as no other training data set could better represent the vertices being classified. The best classification results of the 3D model contained far less noise than the results of classifying the exemplar data set. This suggests a weakness in our choice of meshing algorithm and how it interacts with the discrete curvature calculation algorithm.

Using the SOM trained on the 3D model being classified we investigated how well it generalized to other models of a similar genre using a combination of scale sensitive and invariant features. The SOM generalized well when the scale sensitive features used had a similar mean value. This illustrated the importance of using a wide range of training data when scale sensitive features are being classified by the SOM.

Chapter 4

Segmentation of Free-form Surfaces

Segmenting a 3D surface into regions that share similar properties is the first step in many recognition tasks. This description by parts can then be used for object recognition, indexing and retrieval, defect testing and reverse engineering.

We will investigate the segmentation of triangulated 3D data into meaningful patches using the vertex classification techniques described in Chapter 3 and MacLennan et al. [2006] combined with region labeling which groups vertices that share the same neuron classification. The effect of altering the training parameters will be evaluated in the same order described in Section 3. The SOM topology and connectivity model, torus and hexagon respectively, will remain fixed throughout the experiments. As demonstrated by the results in Chapter 3 these parameters have a significant effect on how the SOM classifies vertices and how well the classifier clusters vertices that share similar curvature properties. We will use the colouring scheme described in Chapter 3 with the addition of lines being drawn around the segmented regions.

The CAD model in Figure 4-1 used in the vehicle manufacturing process was supplied by Dr Bernard Rolfe will be segmented using the SOMs trained with varying parameters, training data and features.

In order to determine what set of parameters produced a reasonable segmentation the following criteria is used:

- Low level of noise in the form of very small regions.
- Well defined surface patches.
- Close approximation to the model's CAD primitives.
- Ability to discern between adjacent regions of different curvature

To augment the visual inspection of the segmentation statistical properties of graphs representing the regions and U-matrices will be used to compare the clustering ability of

SOMs trained with different parameters. We will use a simple algorithm to connect vertices on the triangulated surface that share the same BMU. An U-matrix will be used to visualise the distance of connected neuron in the SOM expressed as the Euclidean distance of their weights.

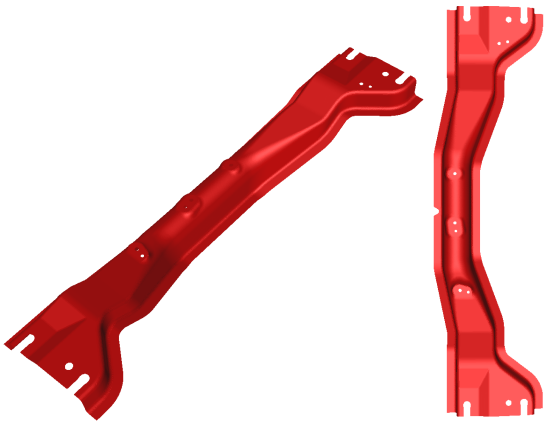
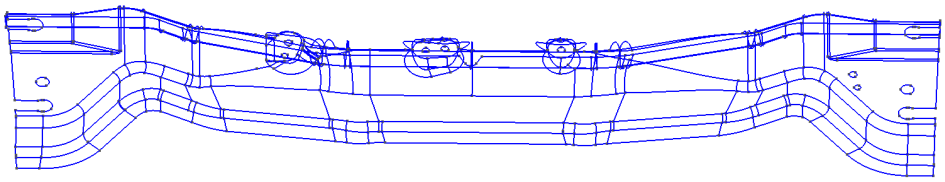


Figure 4-1: Rendered views of CAD Model

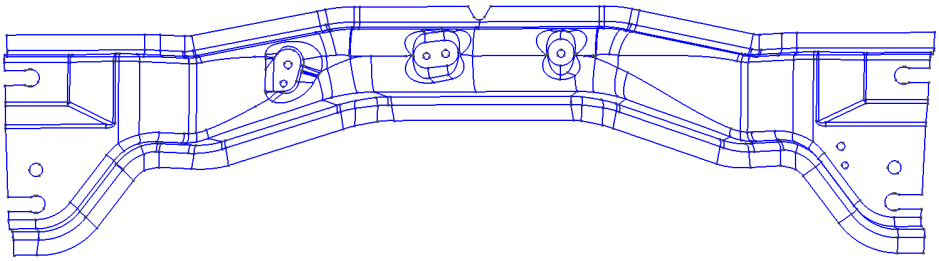
Our patch clustering algorithm converts the triangulated mesh's underlying edge and vertex structure into a generic graph representation augmented with the BMU of each vertex. For each vertex in the graph not already a part of a cluster it performs a depth first search for other connected vertices by traversing between vertices that share the same BMU.

In MacLennan et al. [2006] the rectangular topology and planar neighbourhood model were used which lead to the boundary affect where neurons at the edge of the model are connected to less neurons causing excessive firing of neurons around the edges [Kiang et al., 1997]. In our investigation we used the toroidal neighbourhood model and hexagonal connectivity model which gives all neurons equal connectivity between themselves with no neuron having less connected neighbours than the others This makes the BMU update function consistent for all neurons.

The data used to train the SOM determines how neurons react to different stimuli and how they cluster vertices based on feature input. Any bias in the training data will be



(a) Side



(b) Top

Figure 4-2: Views of the IGES representation of the CAD Model

reflected in the resultant SOM after training. To investigate how this affects the segmentation after classification and connecting vertices that share the same cluster membership we will use the following training data sets:

- Exemplar paraboloids (Figure 4-4 which are the same as the exemplars in Figure 3-1 on page 56)
- CAD model being segmented (Figure 4-1)
- Surfaces generated using a random NURBS generator and meshed using GMSH (Figure 4-3)

These sets of surfaces were chosen as they represented three types of training data: random surfaces, exemplars from a class of surfaces (conics) and the object being segmented, potentially an ideal training data set. We chose to build a large random set of surfaces as training data because there would be no preconceived notion of how the SOM stimuli should be structured. We don't necessarily know what surfaces will be processed by

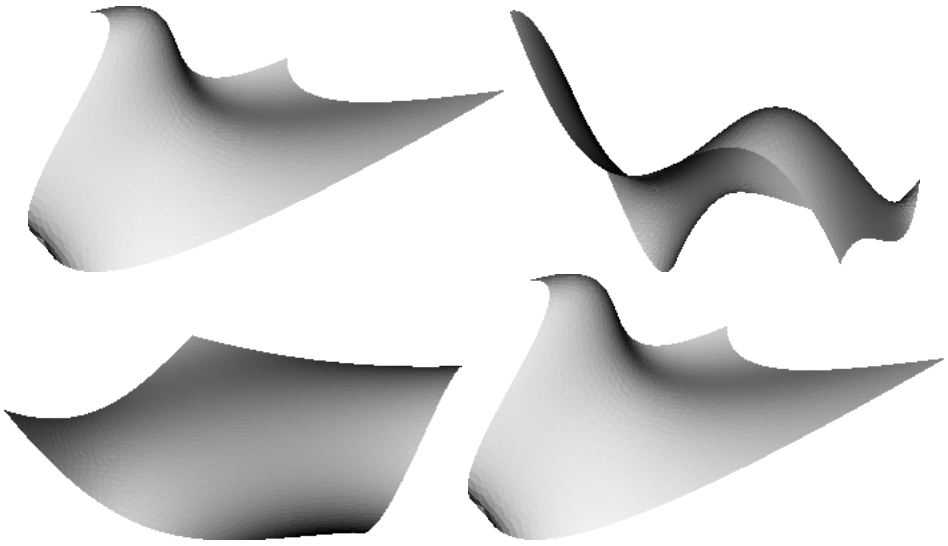


Figure 4-3: Selection of 84 randomly generated free-form surfaces.

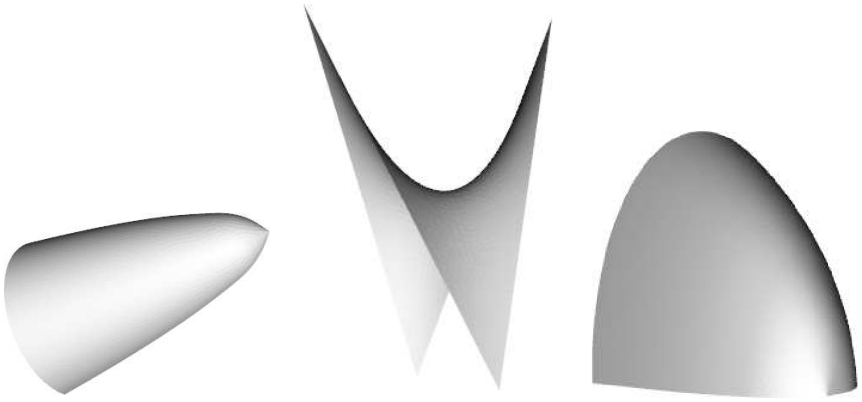


Figure 4-4: Paraboloids.. The functions used to create these surfaces are shown in Table 3.1 on Page 56.

the SOM in the future so building it with many random surfaces may yield good results across many input surface types. Examples of CAD models that describe a wide range of free-form surfaces are car panels, computer mice, boat hulls and teapots. Modern CAD systems can deal with more complex shapes than the regular primitives such as planes, cylinders, toroids, codes and spheres. This is demonstrated by the CAD model shown in Figure 4-2 where sections transition from low k_2 to high k_2 whilst maintianing a constant k_1.

4.1 Paraboloids

Using the training data set described in Section 3.1 shown in Figure 4-4 to train SOMs, we will classify the vertices of the CAD model illustrated in Figure 4-1 to create patches of connected vertices that share the same cluster represented by a neuron in a SOM. This dataset is attempting to map the surface types in CAD models to variations of paraboloids. This dataset represents many combinations of k_1 and k_2 in a compact form that could be found on a wide range of surface types. As with the random surface dataset we didn't want to make too many assumptions about what the CAD model represented as these modelling systems by design can represent many surface types.

4.1.1 SOM Size

For this experiment we used the feature set KH which is the same pair of features used by Besl and Jain [1988] described in Table 2.2 to categorise curvature types. The number of training iterations will be set to one epoch for both fine and coarse training processes. One epoch is the number of vertices in the training data set.

The size of the SOM dictates the maximum number of vertex types that can be represented by this technique. We want our experiments to illustrate a reasonable range of

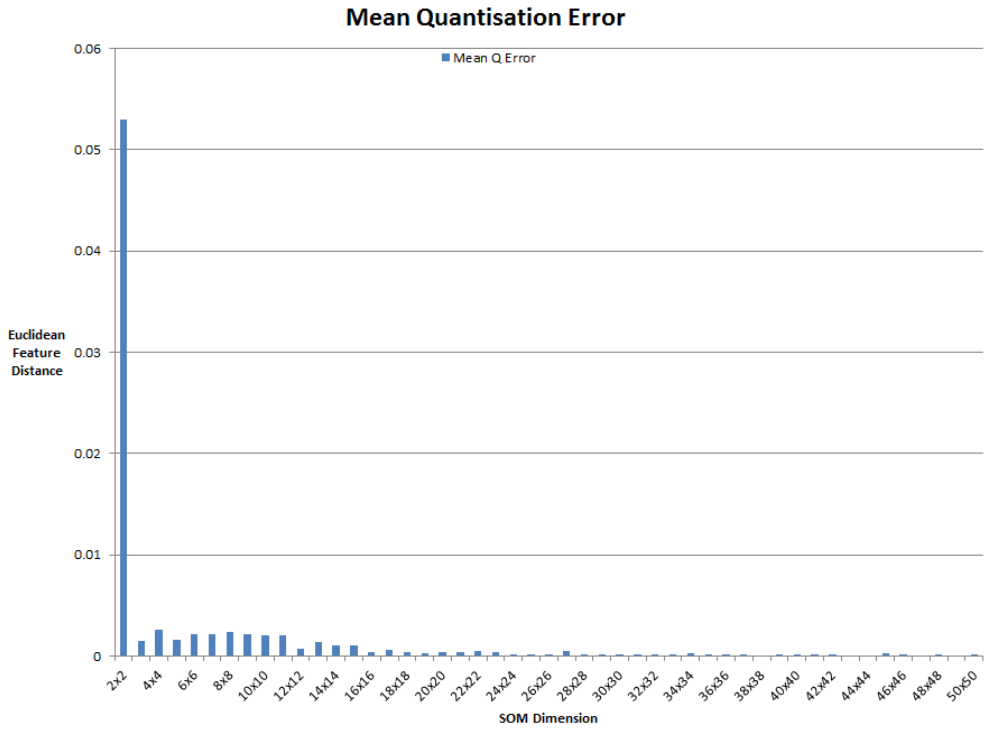


Figure 4-5: Paraboloid data set: Mean Quantisation error when varying SOM dimensions

segmentation results so the size range will be constrained to be between 2×2 and 20×20 inclusive unless the results direct us otherwise.

We will examine changes in Q as the SOM size changes when classifying the vertices of the CAD model using a SOM trained using the paraboloid exemplars. Statistics will be gathered to examine (1) the number of unique patches on the CAD model after the segmentation process has completed, (2) visualisations of BMU distributions for the CAD model's vertices, (3) percentage of neurons fired during the classification process, and how these factors and results are related to the segmentation results.

The initial SOM size of 2×2 (Figure 4-12(a)) is unable to segment any regions, as shown by the single large region of purple. The descriptive plots of mean Q, mean patch area, percentage surface area covered by the largest patch and the percentage of neurons fired during the classification process supports this observation. All of the data plotted was

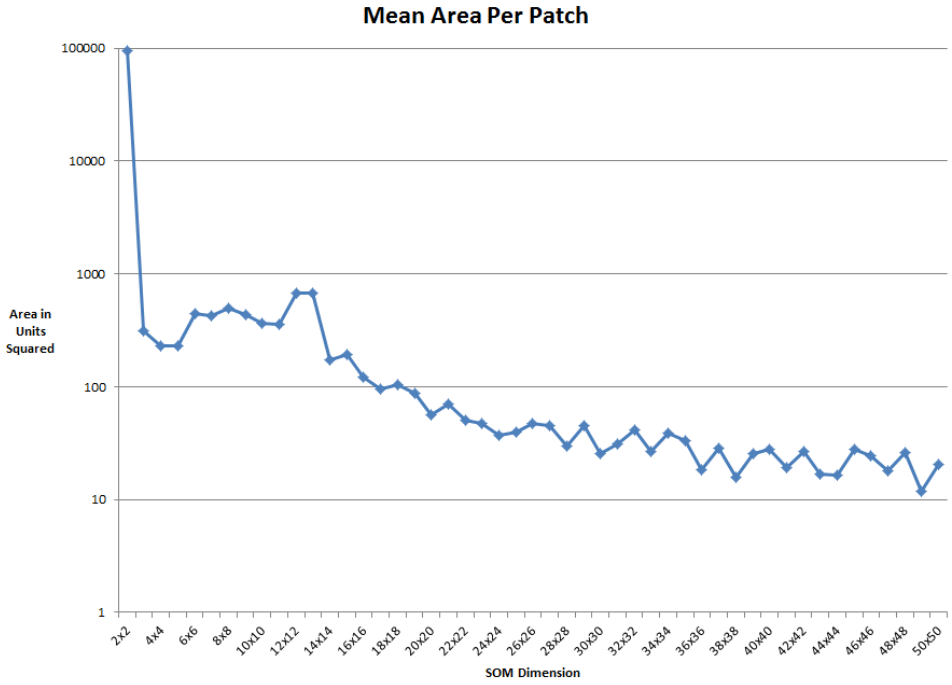
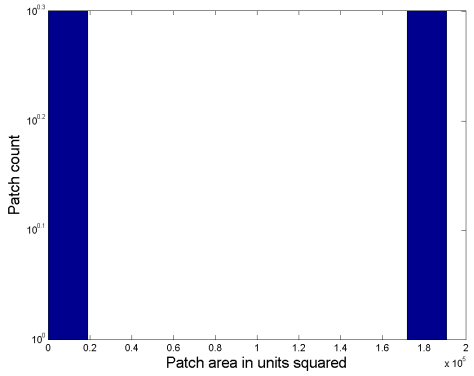
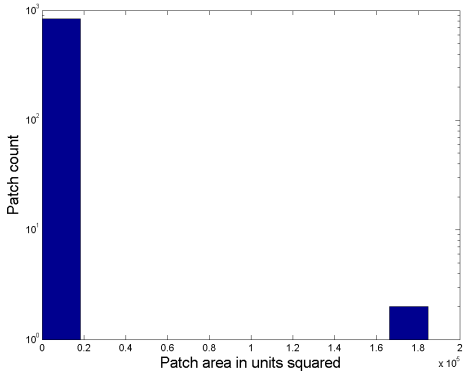


Figure 4-6: Paraboloid data set: Mean Patch area when varying SOM dimensions

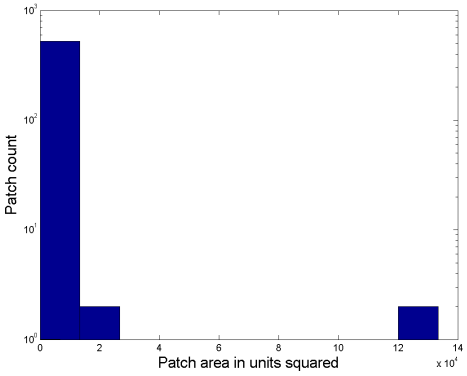
from a single run as we always process the training data in a consistent order to ensure repeatability of our experiments. The Q error drops significantly as the SOM size increases (Figure 4-7). The plot of the mean patch areas by SOM size (Figure 4-6) supports our observation of one region dominating the whole surface with 100000 units^2 of surface area compared to values ranging from 100 to 1000 for the other SOMs. This is confirmed by the patch area histograms in Figure 4-7 where only two patches are present for the 2×2 SOM with one far larger than the other and an increase in patch count and decrease in patch size as the SOM size grows. The distribution plots show the number of outlier patches outside the bucket representing the smallest patches is increasing as they grow. This increase is visible because the maximum patch size is decreasing making the smaller differences between patch sizes visible. The percentage of surface area belonging to the largest cluster (Figure 4-11) reinforces this observation further with close to 100 percent of the surface represented by one neuron for the 2×2 SOM and decreasing until it stabilises at around 22×22.



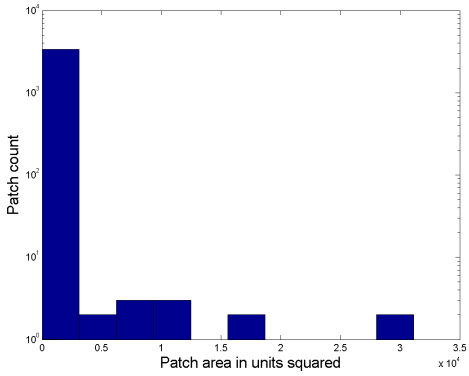
(a) 2×2



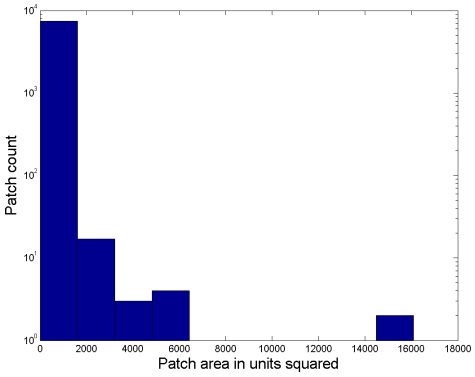
(b) 5×5



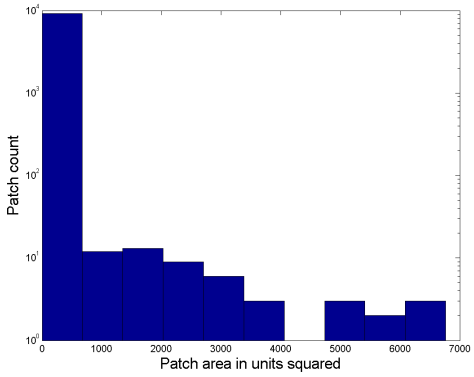
(c) 10×10



(d) 20×20



(e) 30×30



(f) 50×50

Figure 4-7: Paraboloid data set: Patch area distribution when varying SOM dimensions with a log scaled y axis

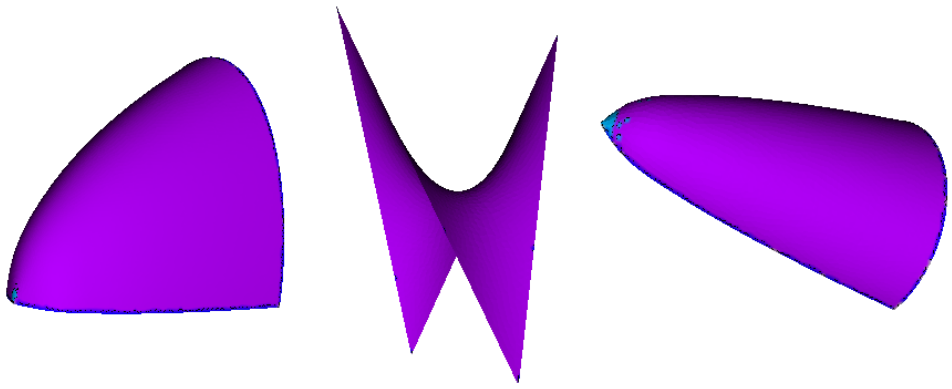


Figure 4-8: Paraboloid data set: Surfaces coloured by BMU for the 2×2 SOM

The results for the 2×2 SOM in Figure 4-12(a) suggests that either the SOM is not large enough to represent the different curvature types or there was insufficient variation in the training data set. The affect of training iterations on the segmentation result will be investigated later in this Chapter. As the size of the SOM grows the number of distinct regions extracted from the CAD model increases. In Figure 4-12(b) the light blue regions along the edges of the model correspond to the convex sections of the training data in Figure 4-9 where the regions are curved in both directions with the curvature in one direction being greater than the other. Small regions are also beginning to grow along the edges of the CAD model where the surface is convex. The segmentation is noisy when k_1 is is larger than k_2 as illustrated by the results of segmenting the paraboloid and strongly curved regions of the elliptic paraboloid. This suggests a weakness in the training parameters or the features KH used in this section. This will be clearer when we use this SOM to segment another object.

When the SOM size is increased we see more clearly defined regions that align with the IGES representation in Figure 4-2. We observed more noise on flat regions on the surface of the CAD model and in previously well segmented regions, such as the convex edge on the lower left and right hand sides for the 15×15 output in Fig 4-12(d) that degenerate in the 20×20 output in Figure 4-12(e).

For each of the results of the segmentation using the paraboloid data set a smaller pro-

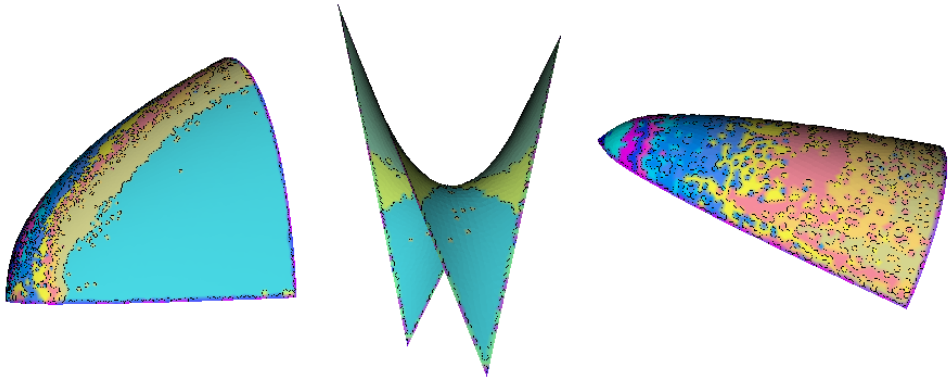


Figure 4-9: Paraboloid data set: Surfaces coloured by BMU for the 5×5 SOM

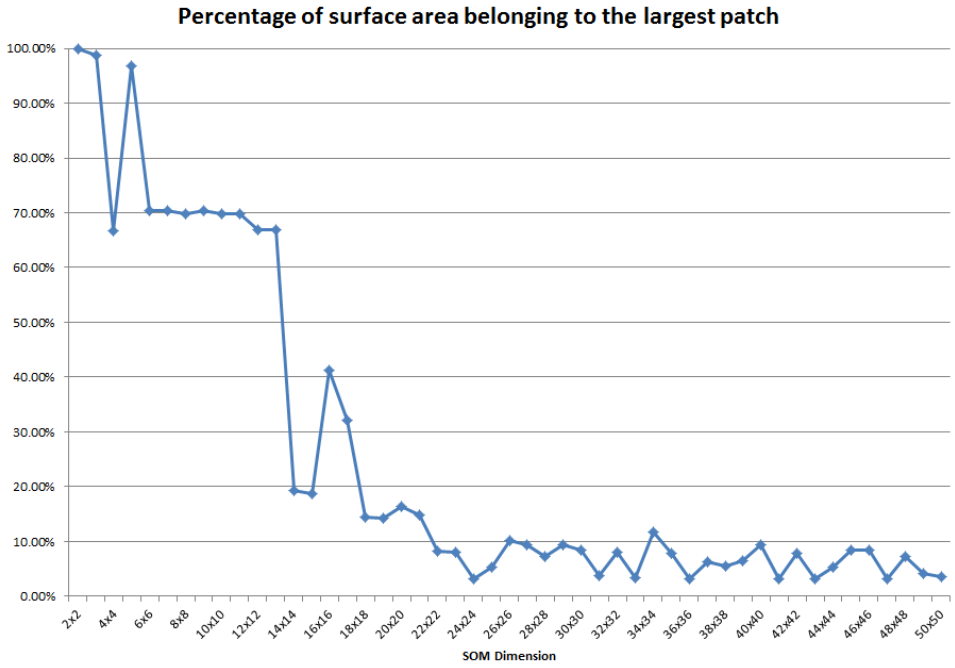


Figure 4-10: Paraboloid data set: Percentage of surface area covered by the largest patch as a function of SOM dimension

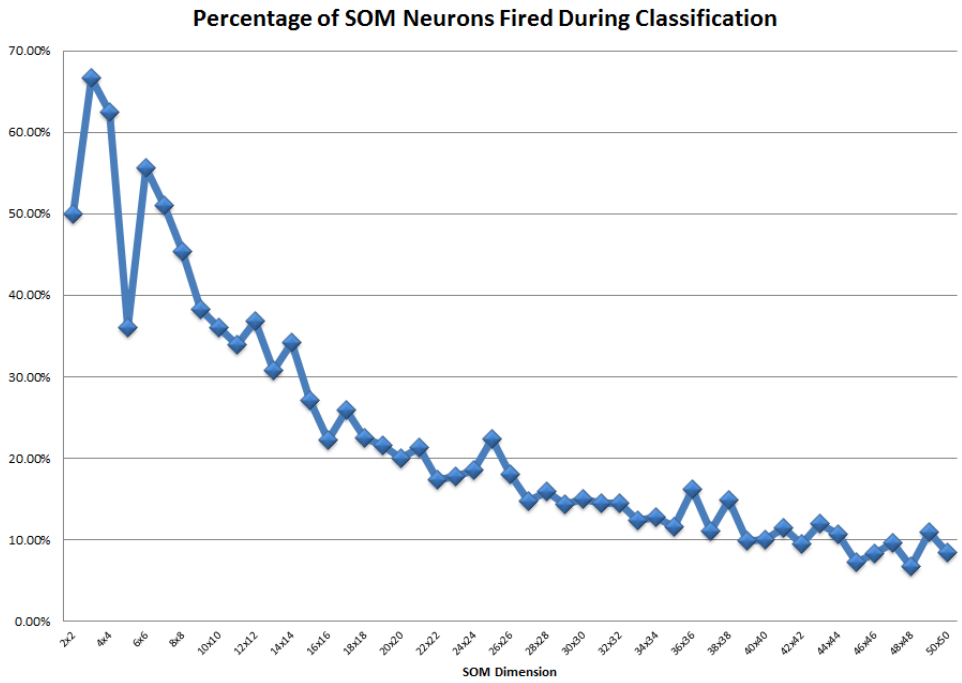


Figure 4-11: Paraboloid data set: Percentage of neurons fired during classification as a function of SOM dimension

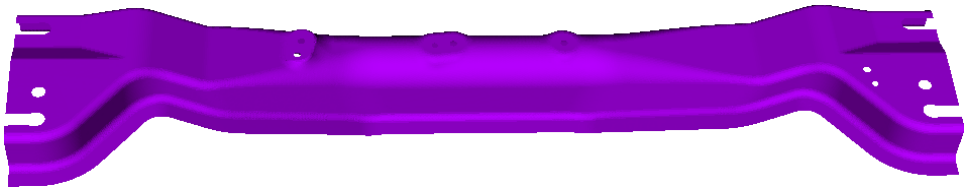
portion of the neurons are fired as the SOM size increased (Figure 4-6) and the mean Q error rapidly dropped once the SOM was larger than 2×2 (Figure 4-5.) When we used the SOM to classify the training data (Figures 4-8 and 4-9) we see that the SOM is able to classify the various parts of the training data set into distinct regions but when applied to the CAD model a small cluster of neurons are fired. This is illustrated by both the percentage of neurons fired in Figure 4-11 and the lack of colour variation when the classification of vertices is projected onto the CAD model being segmented. For larger SOMs similar colours signifies neurons that are close to each other in terms of connectivity on the SOM.

The 15×15 SOM in Figure 4-12(d) resulted in a reasonable balance between distinct regions extracted while not introducing as much visual noise compared to the regions extracted for larger SOMs. However this training data set does not sufficiently represent the feature set H and K on the CAD model.

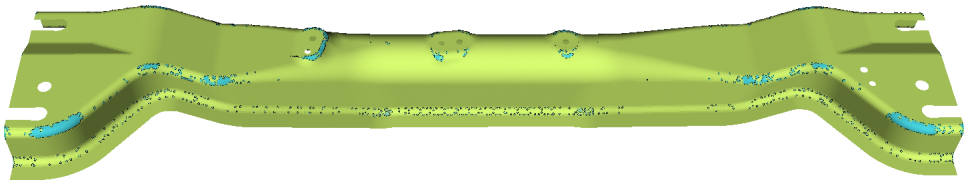
The training data set does not contain enough variation in H and K as illustrated by the plots describing percentage of neurons fired (Figure 4-11) where a smaller subset of the SOM is used when classifying the surface vertices. The mean area per patch is dropping and with this small mean patch size comes noise as shown on the relatively flat regions in Figure 4-12(e). The training data set may do better when using different feature combinations, which we will investigate next.

4.1.2 Feature Selection

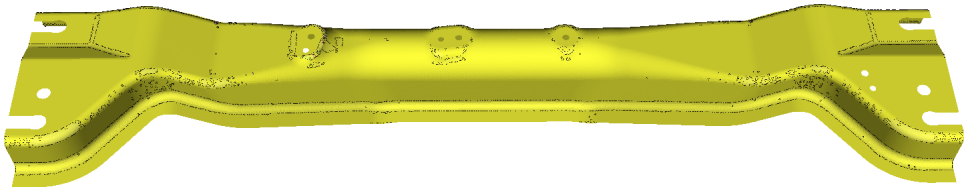
In the previous Section we found that a good balance in detail versus noise when segmenting the CAD model was achieved when: (1) mean area per patch was within the range of 100 to 1000 units squared (Figure 4-13), (2) the percentage of surface area belonging to the largest patch was between approximately 15 to 45 percent (Figure 4-17), and (3) the percentage of neurons fired during classification was between 30 to 15 percent (Figure 4-18.)



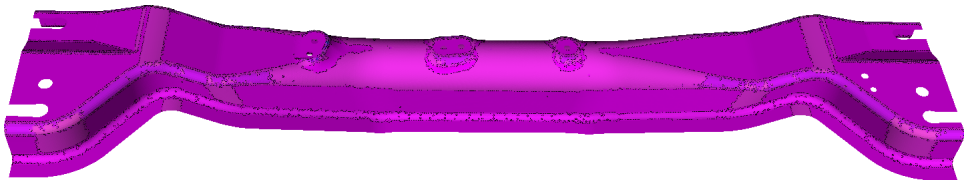
(a) 2×2



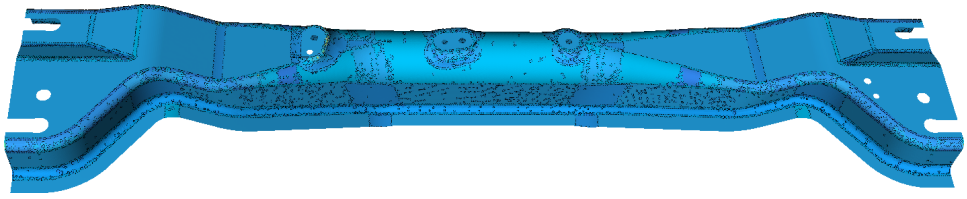
(b) 5×5



(c) 10×10



(d) 15×15



(e) 20×20

Figure 4-12: View of surfaces segmented using a SOM trained on Paraboloids

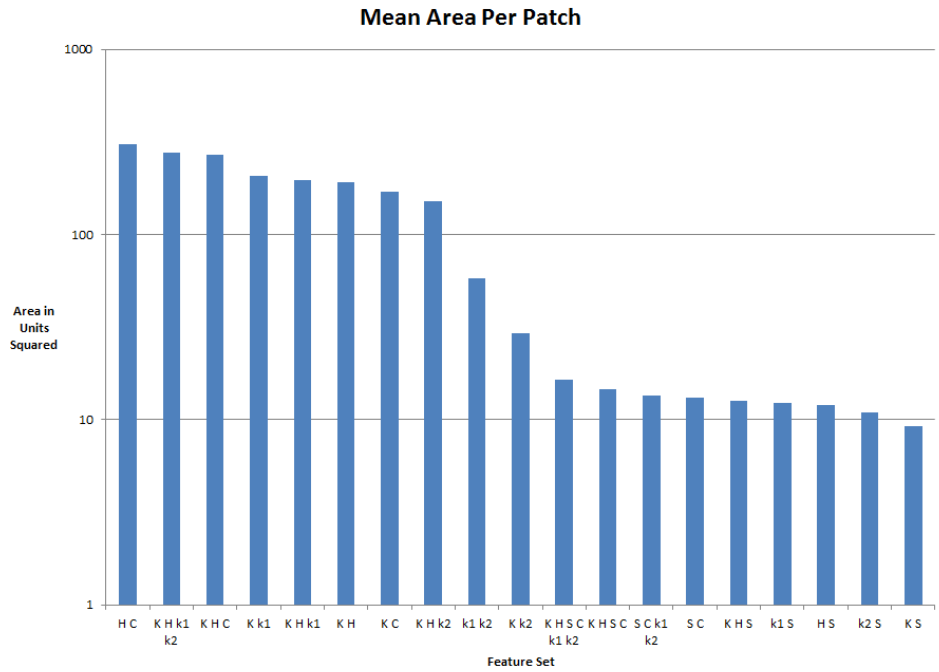
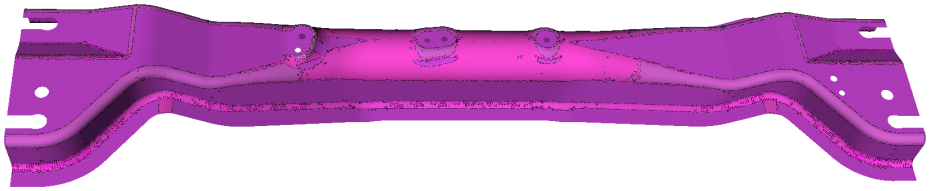


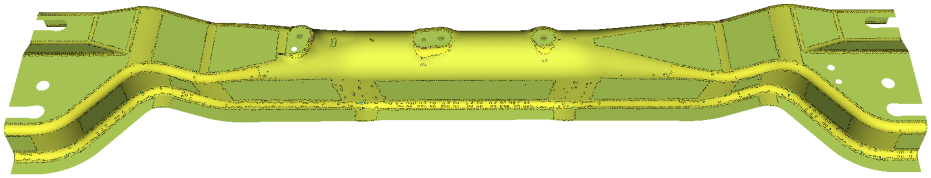
Figure 4-13: Paraboloid data set: Mean Patch area when varying SOM features

Using the results from the previous Section as a starting point we varied the feature set to see if these ranges are an indicator of good results. Q remained relatively constant once the SOM was larger than 15×15 so we will use this as a starting point for choosing feature sets for investigation. We found that the feature sets KHk_1 and HC are within these ranges.

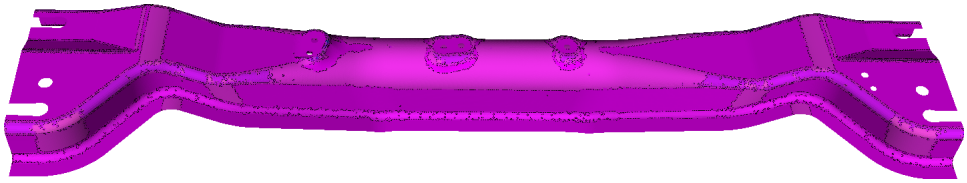
The KHk_1 feature set (Figure 4-14(c)) segments regions that are separated by areas of strong curvature within the model, typically where a flat region meets a curved area that bends strongly in one direction but is unable to detect other features. HC (Figure 4-14(b) does an better job of segmenting the CAD model compared to KH (Figure 4-14(a)) and KHk_1 (Figure 4-14(c)) as it is can segment flat regions that are separated by sections that are curved in one direction only, as illustrated by the regions shown in Figure 4-15. It can detect regions that are gently concave and convex when there is little or no curvature orthogonal to the direction of the curvature of greatest magnitude. One weakness of the HC feature set compared to the other two is that it can't extract regions where there is



(a) KHk_1

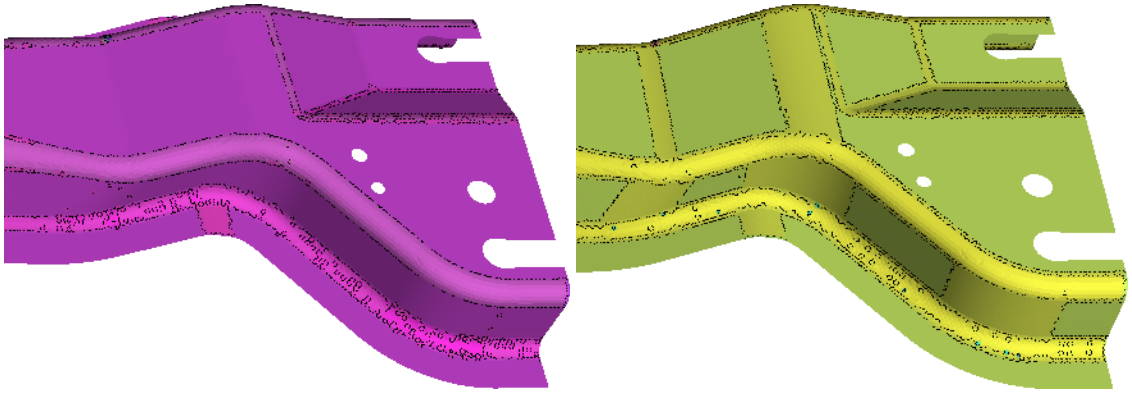


(b) HC

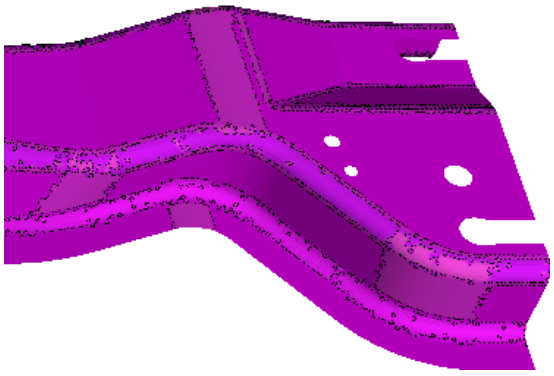


(c) KH

Figure 4-14: Paraboloid data set: Feature sets that fell within range of mean area per patch, percentage of area covered by the largest patch and percentage of neurons fired during classification derived from Section 4.1.1

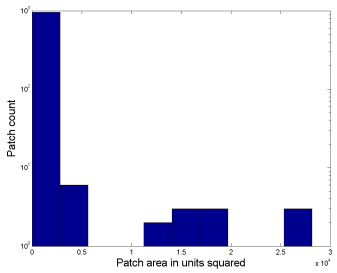


(a) KHk_1 (b) HC

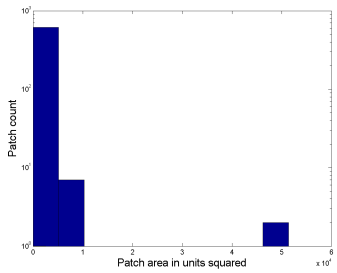


(c) KH

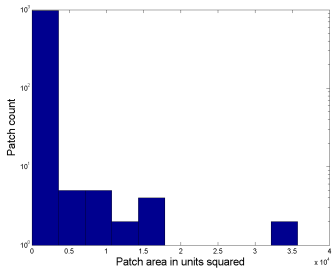
Figure 4-15: Paraboloid data set: Close up view of segmented CAD model from Figure 4-14



(a) KHk_1



(b) HC



(c) KH

Figure 4-16: Paraboloid data set: Patch size distributions with a log scaled y axis for the results in Figure 4-14

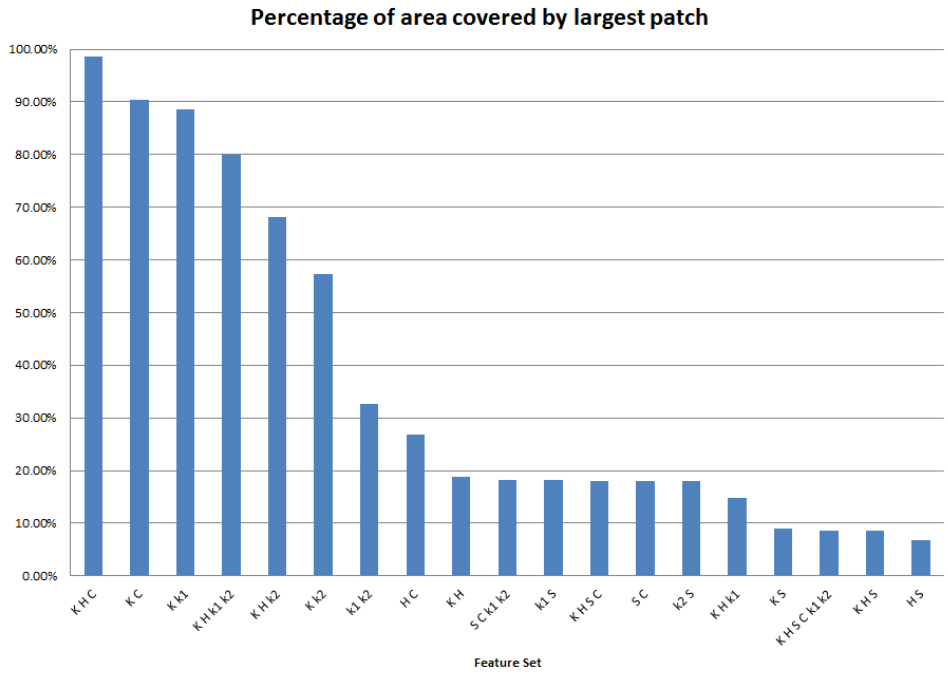


Figure 4-17: Paraboloid data set: Percentage of surface area covered by the largest patch when varying SOM features

a large magnitude of curvature in one direction with the orthogonal curvature moving from 0 to a small positive or negative value. Both feature sets that contain KH show similar distributions of patch sizes and a greater number of patches compared to the results using HC which has less patches and a larger proportion of patches with an area greater than 0.1 (Figure 4-16.)

We will now examine the segmentation results of the feature combinations which lie outside of the range of properties derived from the previous Section and determine if there is a relationship between the value of each of the properties and how the CAD model is segmented. These features are:

- Mean area per patch: Max:HC Min:KS
- Percentage of area covered by the largest patch: Max:KHC Min:HS

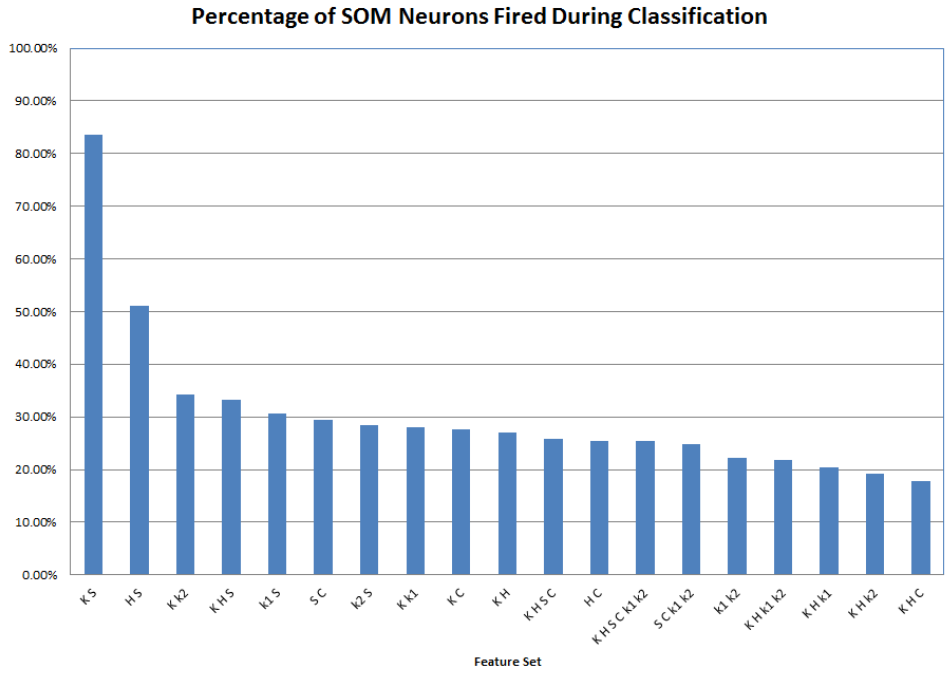


Figure 4-18: Paraboloid data set: Percentage of neurons fired during classification when varying SOM features

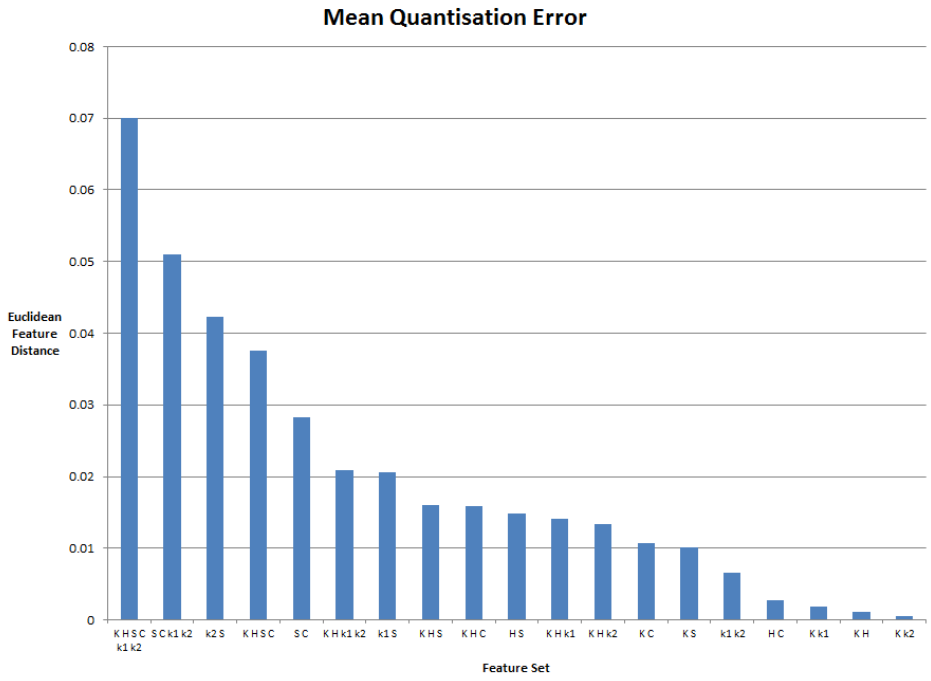


Figure 4-19: Paraboloid data set: Mean Quantisation error when varying SOM features

- Percentage of neurons fired during classification: Max:KS Min:KHC
- Mean Q error: Max: $KHSCk_1k_2$ Min:Kk_2

When comparing the segmented regions of HC (Figure 4-14(b)) and KS (Figure 4-20) we can see immediately there is too much noise for KS where regions that should be represented as a contiguous block, such as the flat triangular regions at the $\frac{1}{3}$ and $\frac{2}{3}$ points on the horizontal axis of the CAD model.

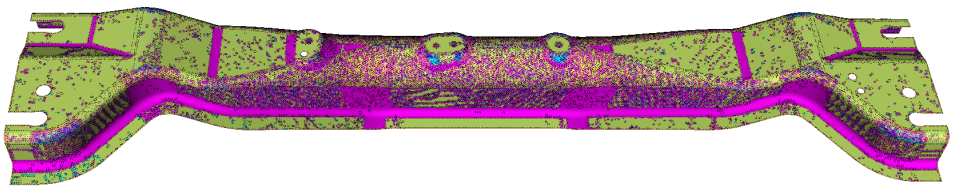
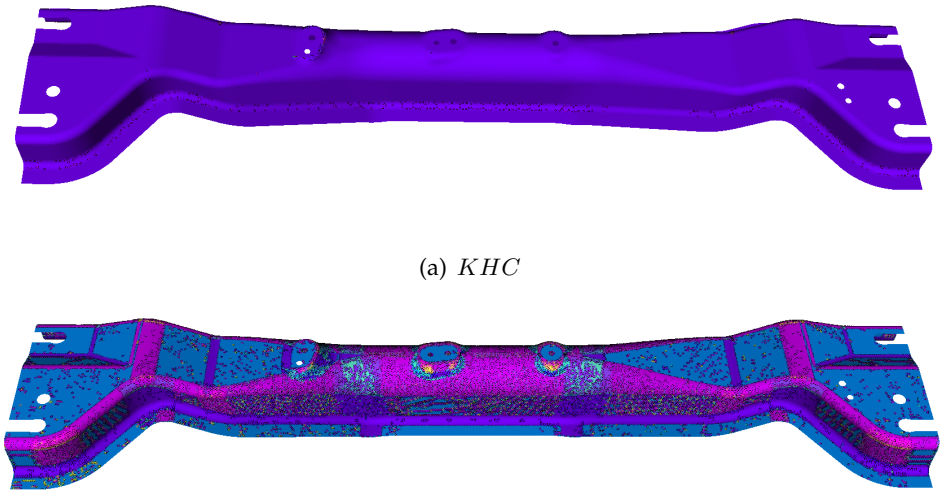


Figure 4-20: View of surfaces classified and segmented using a 15×15 SOM trained on Paraboloids using the feature set KS



(a) KHC

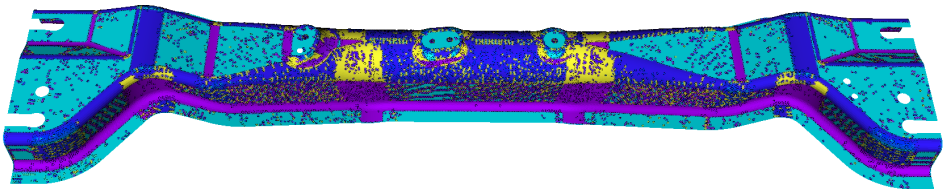
(b) HS

Figure 4-21: View of the surfaces that have the largest and smallest percentage of the CAD model surface area covered by single segmented region

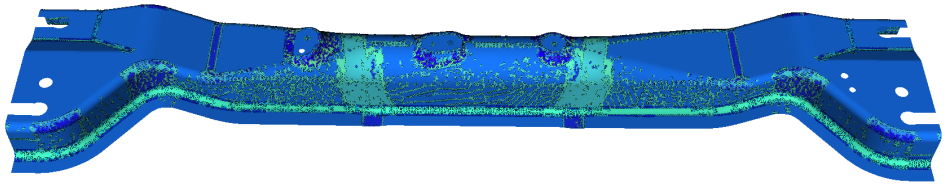
The feature sets KHC and HS produced the largest and smallest values for percentage of surface area covered by a single patch (Figure 4-21) and illustrates how this value can be used to identify which feature sets produce a balanced result. When the value is too large a single patch dominates the surface and too small suggests that the segmentation is noisy. This is assuming that the surface being segmented isn't a single region of consistent curvature. The shading on this model that could be interpreted as separate regions is a result of our rendering technique designed to highlight the shape of the object in 3D.

By segmenting the CAD model with each feature combination we could present the results of the feature set that is the approximate mid-point of this measure and use a binary search approach to direct the user to a satisfactory segmentation with little interaction. Alternatively we can at least target a value for this measure while using other optimization techniques to determine SOM size.

The smallest and largest values for percentage of neurons fired during classification KS and KHC (Figures 4-20 and 4-21(a)) don't tell us as much in terms of segmentation result. Depending upon the surface being segmented we could have a reasonable number



(a) $KHSCk_1k_2$



(b) Kk_2

Figure 4-22: Feature sets with the largest and smallest Q error for the Paraboloid training data set

of regions extracted from the model when the percent fired is low or high, but if the percentage of neurons fired during classification is biased to only one or two neurons then we will have a poor result. This assumes the model is reasonably complex with more than one region of constant curvature.

When the Q error is at its maximum (Figure 4-22(a)) the BMU colouring of each segmented region shows a greater spread across the SOM (Figure 4-23) compared to the smallest Q error (Figure 4-22(b).) This could be a reflection of the training data set and is not necessarily a reflection of the quality of the segmentation. The maximum Q error does segment the regions into something that approximates the underlying IGES model in Figure 4-2 but the result is too noisy.

The feature variation discussed in this Section produced segmented regions varying from sparse (KHC), noisy ($KHSCk_1k_2$) and good (HC). We will use these feature sets when investigating how training iterations affect the segmentation results.

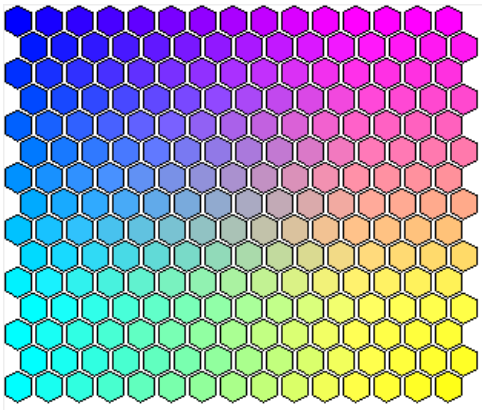


Figure 4-23: 15×15 SOM BMU colour map

4.1.3 Training Iterations

To investigate the effect that training iterations have on the segmentation of our CAD model we chose three feature sets from the previous Section that produced sparse, noisy and good segmentation results and then varied the number of fine and coarse training epochs in the same way we did in Section 3.1.4.

In the previous Section the feature set KHC produced a poor segmentation with a single patch when using 1 coarse and fine training epoch. By varying the fine and coarse training iterations we found the mean patch size didn't vary greatly except for the case where there were no fine or coarse training iterations as shown in Figure 4-24. In the case of no fine and coarse training epochs the mean area per patch increased by an order of magnitude, making what was a poor segmentation for 1 coarse and fine training epochs even worse.

Increasing both coarse and fine training iterations marginally improved the segmentation (Figure 4-25) but the feature set KHC regardless of training iterations does not segment the CAD model as well as the HC feature set. Ignoring the artefacts from the shading the main observation was the light green curved regions have been extracted but we failed to segment the planar sections and other curved regions. The linear initialization doesn't

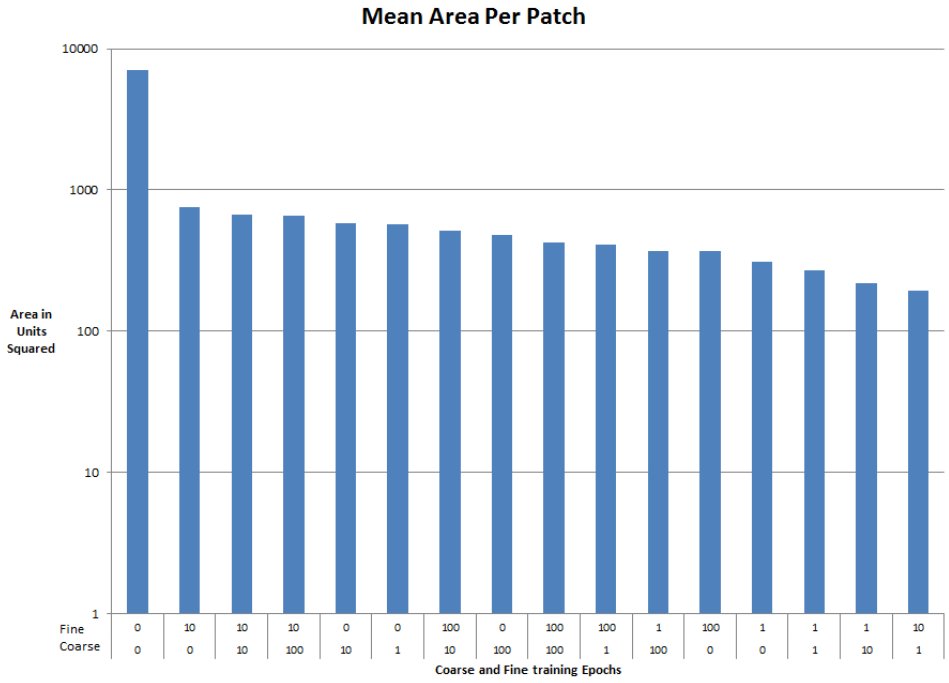


Figure 4-24: Mean area per patch KHC iteration variation

perform well as a bucketing function because there are vertices with curvature values far greater than the majority of the data set that skews the function so 99 percent of the vertices belong to the same neuron. This broad range of curvature values also affects the model when a large number of training iterations is used for this feature set. Each of these curvature metrics measure magnitude compared to S and the mappings described by Besl [1988] which can cause the initialization of the SOM to reduce its usefulness as a classifier.

The noisy feature set of $KHSCk_1k_2$ show in Figure 4-22(a) was initially trained using 1 coarse and 1 fine training epochs and produced a noisy segmentation. When the number of coarse and fine training epochs was reduced to 0 the mean patch size was greater than for any of the other training iteration combinations resulting in a segmentation with less noise on the curved regions. It was unable to reduce the level of noise (very small segmented regions) on the flatter sections. When the training iterations were set to the maximum for the experiment (100 epochs) the level of noise was at its greatest for the

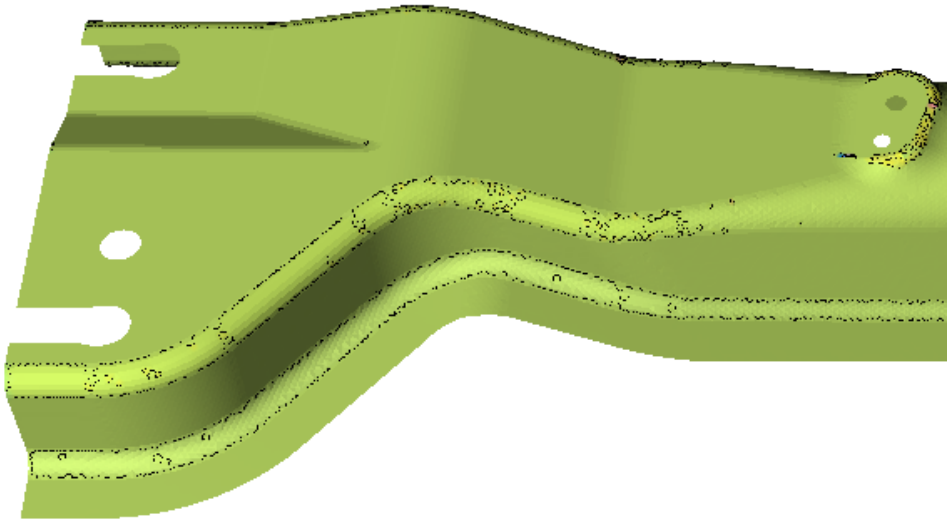


Figure 4-25: Close view of CAD model segmented using SOM trained with KHC and 100 coarse and fine training epochs

feature set $KHSCk_1k_2$.

The mean patch size histogram in Figure 4-26 shows that for this feature set the coarse training iterations is the main factor that determines how small the mean patch is. The coarse iteration count shown on the top row of values on the x-axis tend to be grouped together, the fine epoch counts don't appear to have a significant influence on the mean patch area. When the coarse and fine training epochs are both set to 0 the mean area per patch is at its maximum which is the same for the KHC feature set shown in Figure 4-24. The classifier output for the maximum and minimum training epochs in Figure 4-27 shows that the 100 fine and coarse training epochs resulted in a SOM that could more identify more detail than just the simple the concave and convex regions that the linearly initialized SOM could detect. The linearly initialized SOM (Figure 4-27(a)) did a better job of bucketing the curvature metrics for this large feature set compared to KHC which resulted in no regions being identified. This can be explained by the greater number of curvature metrics that appear to be balancing out the extreme values in some of the feature sets.

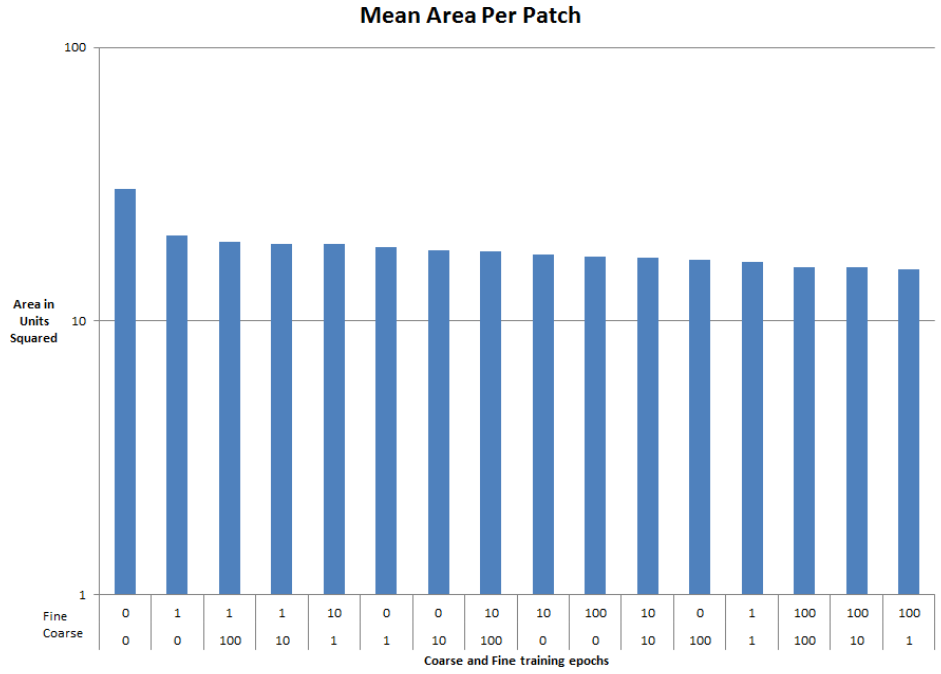
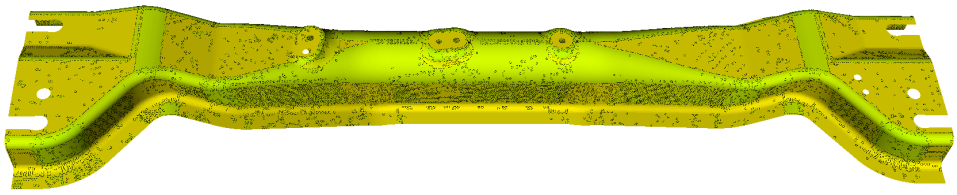
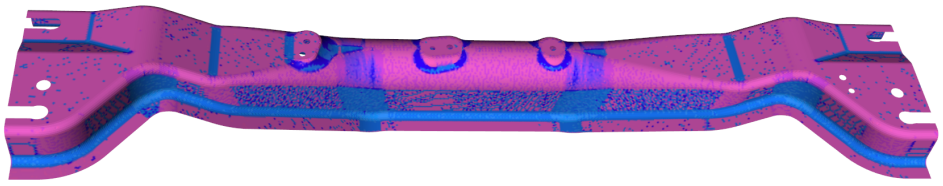


Figure 4-26: Mean area per patch $KHSCk_1k_2$ iteration variation with coarse iterations on the upper row and fine iterations on the lower row.

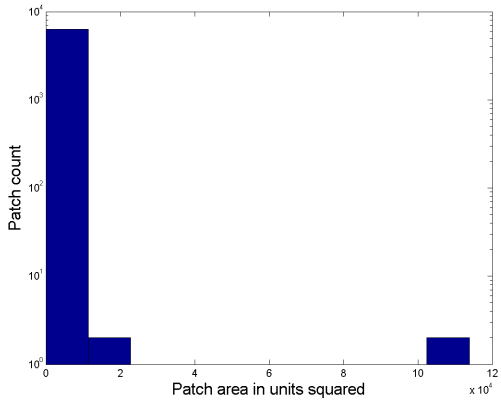


(a) 0 coarse and 0 fine training epochs.

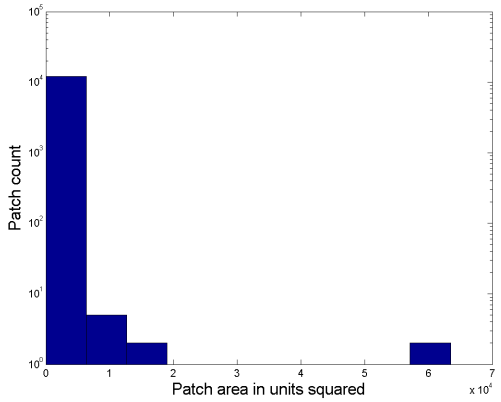


(b) 100 coarse and fine training epochs.

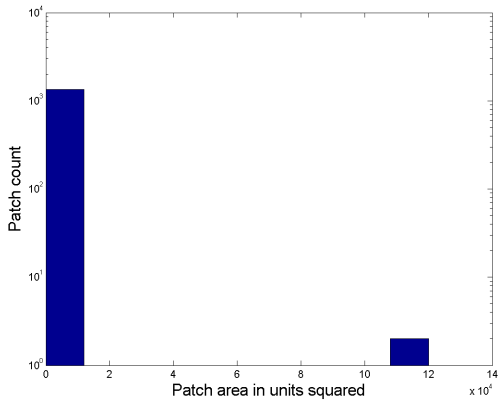
Figure 4-27: CAD model segmented using SOM trained with $KHSCk_1k_2$



(a) $KHSCk_1k_2$ 0 coarse and 0 fine training epochs.



(b) $KHSCk_1k_2$ 100 coarse and fine training epochs.



(c) HC 10 coarse and 100 fine training epochs.

Figure 4-28: Paraboloid data set: Patch size distributions with a log scaled y axis for the results in Figures 4-27 and 4-29

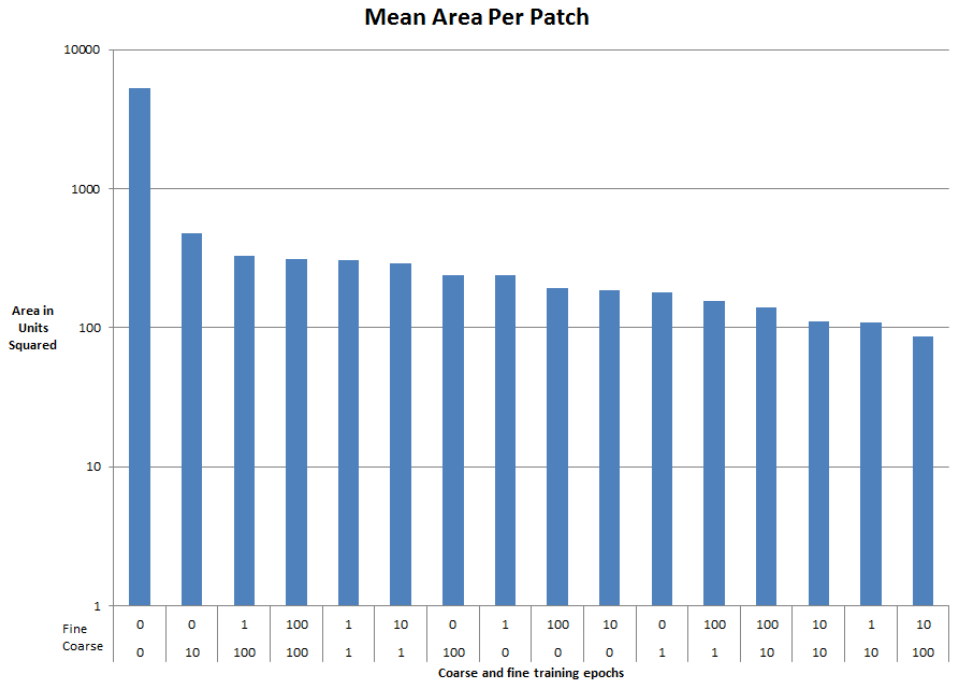


Figure 4-29: Mean area per patch HC iteration variation

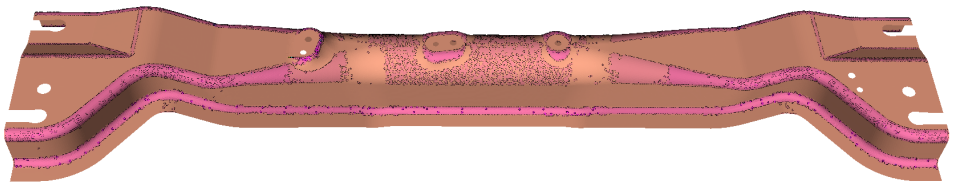


Figure 4-30: CAD model segmented using SOM trained with the paraboloid data set and features HC with 10 coarse and 100 fine training epochs.

The least noisy results for the 15×15 SOM with one coarse and fine training epoch trained with the paraboloid data set and features HC. We varied the coarse and fine training iterations and found that the maximum mean patch size (Figure 4-29) was generated when both iterations were set to 0. This result was the same for the other feature sets examined in this Section. This is happening because the SOM isn't being trained and is instead being used as a bucketing function with fixed intervals to classify the vertices. Our tecnique cannot be used as a bucketing function because extremes in feature values can cause a large proportion of the vertices to be put in the same bucket. Training the SOM alters the neuron weights to better match the distribution of feature values.

The training iteration counts of 10 coarse and 100 fine training epochs that generated the smallest mean patch size is not the same as for the other two feature sets discussed in this Section and is not the best iteration combination in terms of segmentation. The results for 10 coarse and 100 fine training epochs shown in Figure 4-30 are poor compared to the segmentation result using 1 epoch for both coarse and fine. In the case of this feature set the smallest mean patch size didn't identify the best result, only the result with the most noise. The noise discussed in 3.3 has affected the segmentation of the CAD model suggesting a limitation with this set of exemplars and their meshing parameters.

After iterating through the feature sets, SOM size and training iterations we found that the most promising results were achieved using a 15×15 SOM with no training. This suggests that the paraboloid training data set doesn't represent a varied or appropriate enough set of surface types and we are better off using the SOM as a bucketing function. The patch size distributions shown in Figure 4-28 illustrates the difference between using the paraboloids as training data and using the SOM as a bucketing function. The feature sets HC and $KHSCk_1k_2$ resulted in a smaller number of patches and a larger mean patch size but none of these combinations produced satisfactory results.

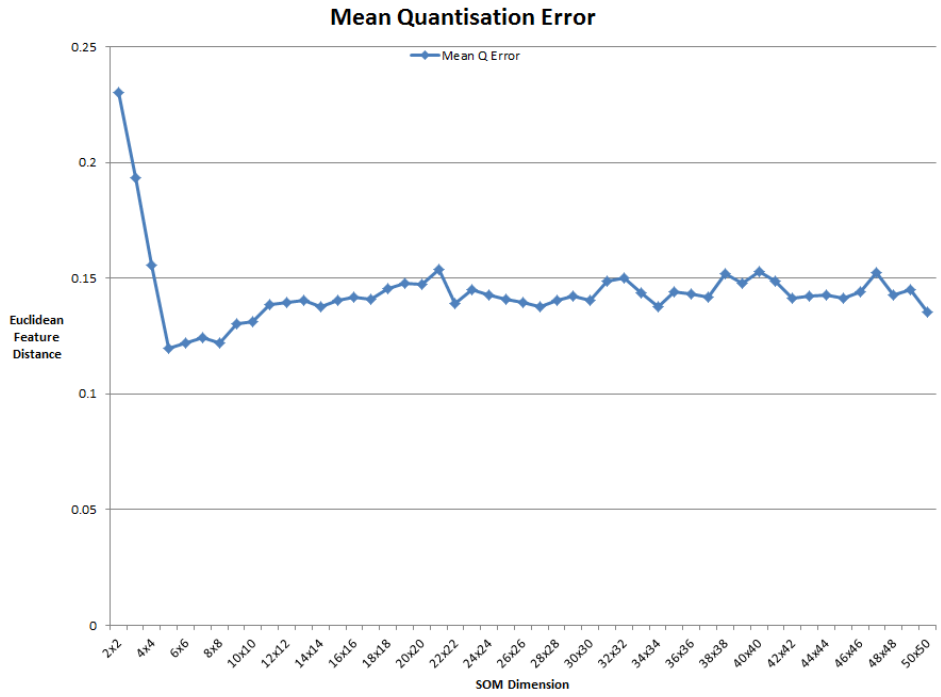


Figure 4-31: Random surface data set: Mean Quantisation error when varying SOM dimensions

4.2 Random Free-form Surfaces

4.2.1 SOM Size

Using the random free-form surfaces shown in Figure 4-3 and the KH feature set we followed the same process described in Section 4.1.1 to identify a SOM size that will produce a good segmentation for the KH feature set with the training iterations fixed at the number of training features in the random free-form surface data set (one epoch.)

The mean Q error (Figure 4-31) drops quickly as the SOM size increases and then stabilises at around 10×10.The stabilisation of Q error as the SOM size increases for the random free-form surface training data set follows the same pattern as the paraboloid data set in Figure 4-5. The mean area per patch stabilises at around the same point as

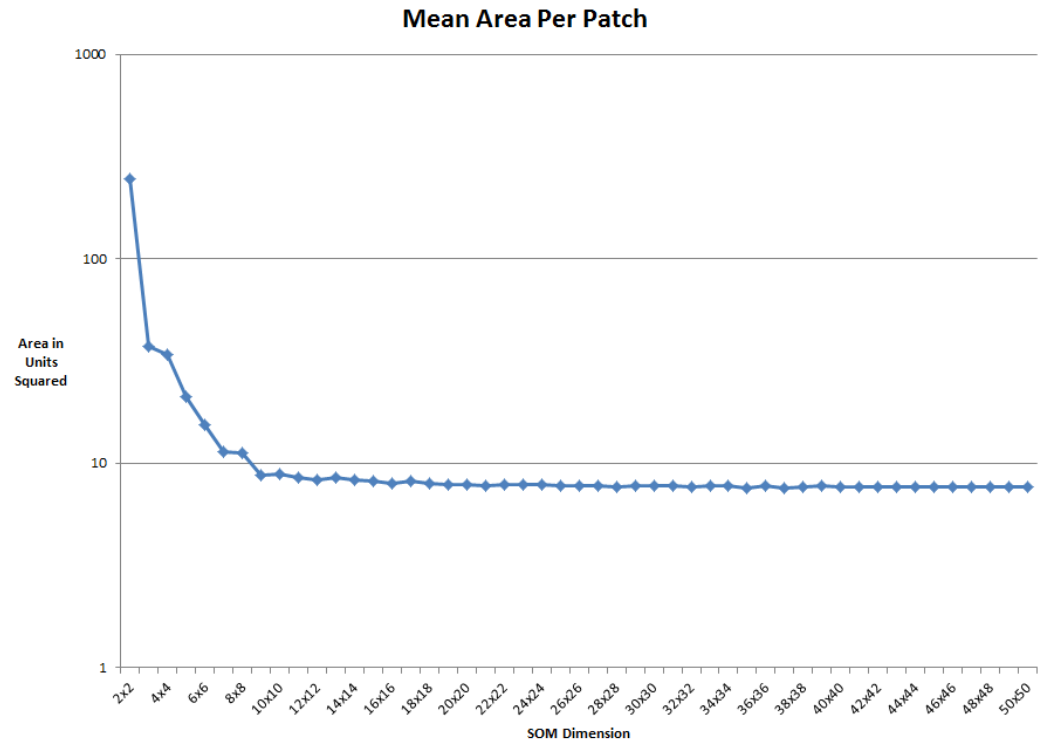
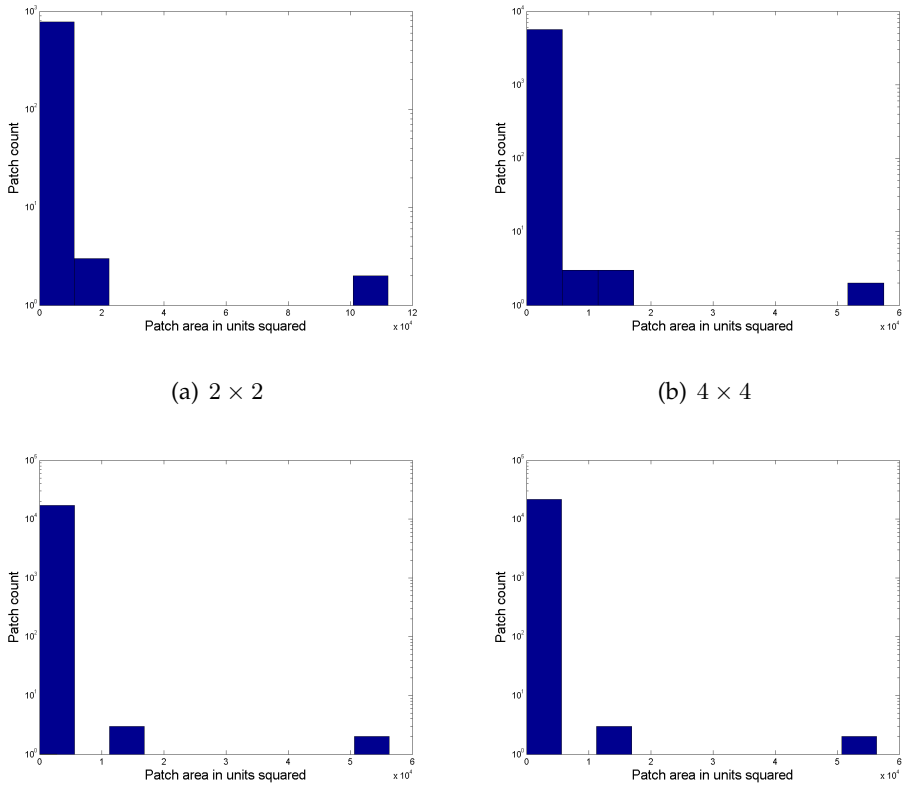


Figure 4-32: Random surface data set: Mean Patch area when varying SOM dimensions

the mean Q error and converges on a value at a greater rate than the Paraboloid data set (Figure 4-6). The percentage of surface area covered by the largest patch (Figure 4-35) is stable within only a few increments of the SOM size compared to the paraboloid data set (Figure 4-10)

Each of the statistical properties shown in Figures 4-31, 4-32 and 4-35 for the CAD model, when segmented using the SOM trained using the random free-form surfaces converges and stabilises at a faster rate compared to the paraboloid data set. The patch size distribution plot in Figure 4-33 shows a significant increase in the number of small patches as the SOM size increases beyond 4×4 which agrees with the plot of mean area per patch (Figure 4-32.)

The paraboloid data set is a set of only three surfaces generated from three functions (Figure 3-1) compared to the set of 85 randomly generated free-form surfaces. This dif-



(a) 2×2 (b) 4×4

(c) 8×8 (d) 10×10

Figure 4-33: Random surface data set: Patch area distribution with a log scaled y axis

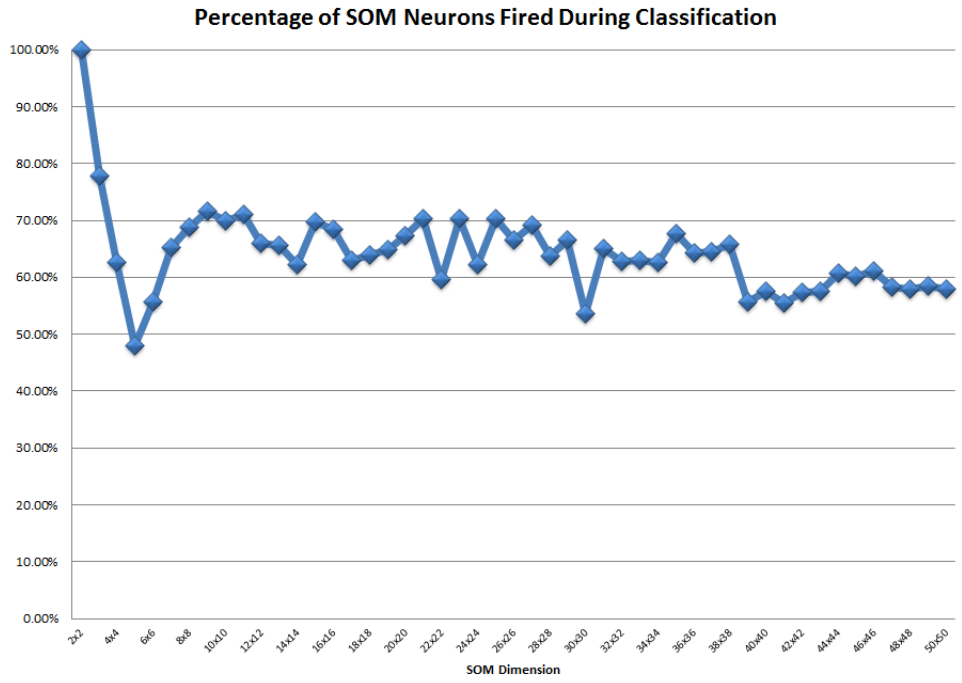


Figure 4-34: Random surface data set: Percentage of neurons fired during classification when varying SOM dimensions

ference is evident in each of the properties that we have plotted. Compared to the results in Section 4.1.1 on page 108 the percentage of neurons fired is greater. The mean Q error stabilises at a higher value and the mean patch size stabilizes at a lower SOM size. Each of these results suggests that the random surface dataset is a better training data source.

The percentage of neurons fired during the classification of the CAD model for the random surface dataset (Figure 4-34) is consistently higher than the percentage of neurons fired during classification for the paraboloid data set in Figure 4-11. This suggests that the random surface data set better represents the variances in curvature metrics on the CAD model. We will continue to compare the percentage of neurons fired across the training data sets throughout this Chapter to see if our hypothesis holds.

The output for 10×10 SOMs trained using the paraboloids and random surface data set are shown in Figure 4-36. The CAD model segmented using the SOM trained on

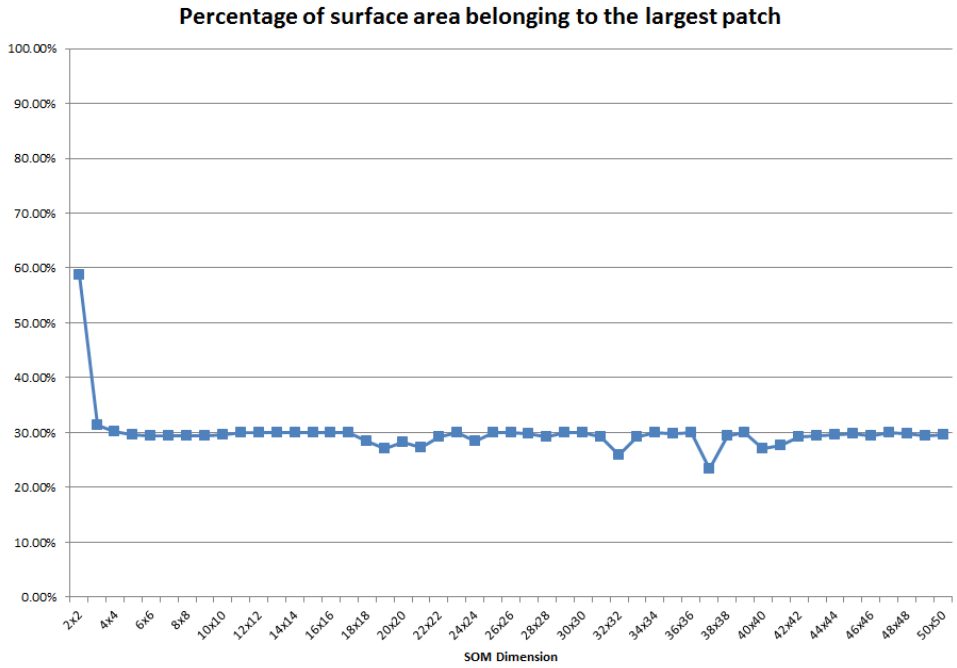
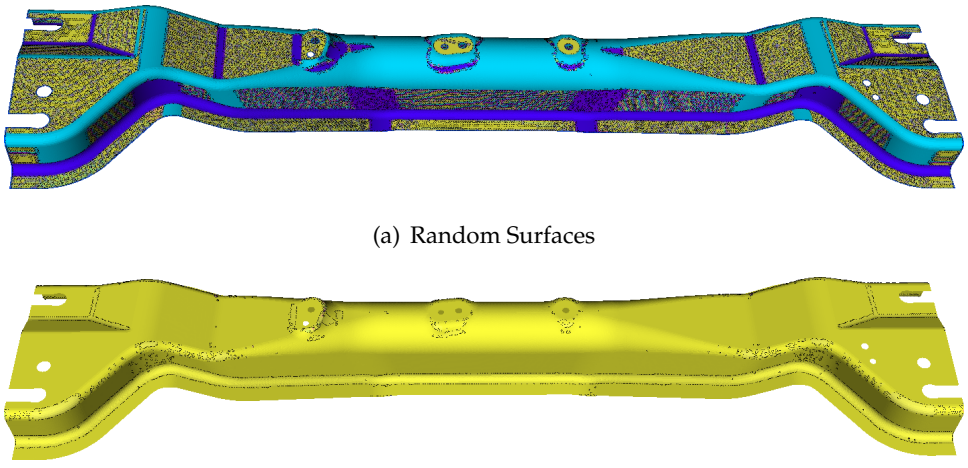


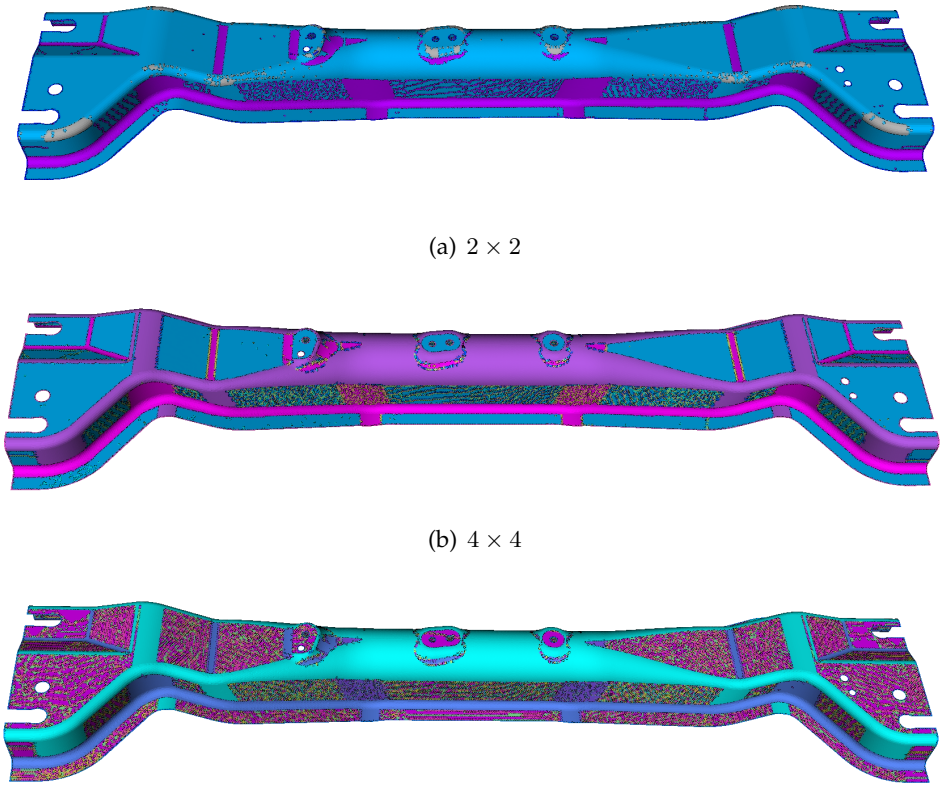
Figure 4-35: Random surface data set: Percentage of surface area covered by largest patch when varying SOM dimensions



(a) Random Surfaces

(b) Paraboloids

Figure 4-36: CAD model segmented using a 10×10 SOM with different training data sets with the features KH



(a) 2×2

(b) 4×4

(c) 8×8

Figure 4-37: Segmentation results for KH feature set trained using random surfaces while varying SOM size

paraboloids (Figure 4-36(b)) shows almost no colour variation, indicating only a small number of neurons fired during the classification process. Few segmented regions that matched the IGES representation of the CAD model in Figure 4-2. When the random surface data set is used the resultant segmentation (Figure 4-36(a)) extracts a greater number of regions with more neurons being fired during the classification process. This is illustrated by the greater number of colours visible on the model. The segmentation is not perfect as planar regions are broken into many tiny regions. Ignoring the noisy segmentation of planar regions, the random surface set produces a better segmentation compared to the output for the paraboloid data set.

The random free-form surface training data set is able to segment the CAD model into

regions that closely matches the underlying CAD model using a smaller SOM compared to the paraboloid training data set. The output in Figure 4-37 shows the progression of segmentation as the SOM size increases with no major changes in regions beyond the 4×4 output in Figure 4-37(b). For each of the segmentation results there is a noticeable amount of noise on the near-planar regions on the sides of the CAD part. This may be a feature of the distribution of KH in the training data set rather than the iteration count and we will investigate this further in the next Section.

4.2.2 Feature Selection

When two SOMs of different sizes are trained using the same data set and feature combinations the mean patch size will likely be larger for the smaller SOM as there are a fewer number of possible clusters. We used the same training parameters and features for each of our training data sets when determining which SOM size to use to evaluate the effectiveness of different features so we can compare segmentation results and their associated measurements.

The mean area per patch when varying the feature set using a 4×4 SOM for the random surface feature set (Figure 4-38) is on average smaller than that of the paraboloid data when training a 15×15 SOM while varying the feature set (Figure 4-13).

The feature set HS produces the smallest mean area per patch (Figure 4-39(a)) and we can see this in the form of a large number of patches on planar and near-planar regions on the segmented CAD model. To determine if the relatively poor segmentation of planar regions is a property of the training data set or the features HS, we examined the segmentation result of the features KHC which on average produced the largest regions (Figure 4-39(c)). We found there was an improvement in segmenting planar regions but there was still many small regions on the sides of the model that should be considered as one region. We also examined the segmentation of the feature set SC (Figure 4-39(b)) which produced a mean patch size which was about the middle of the range of mean

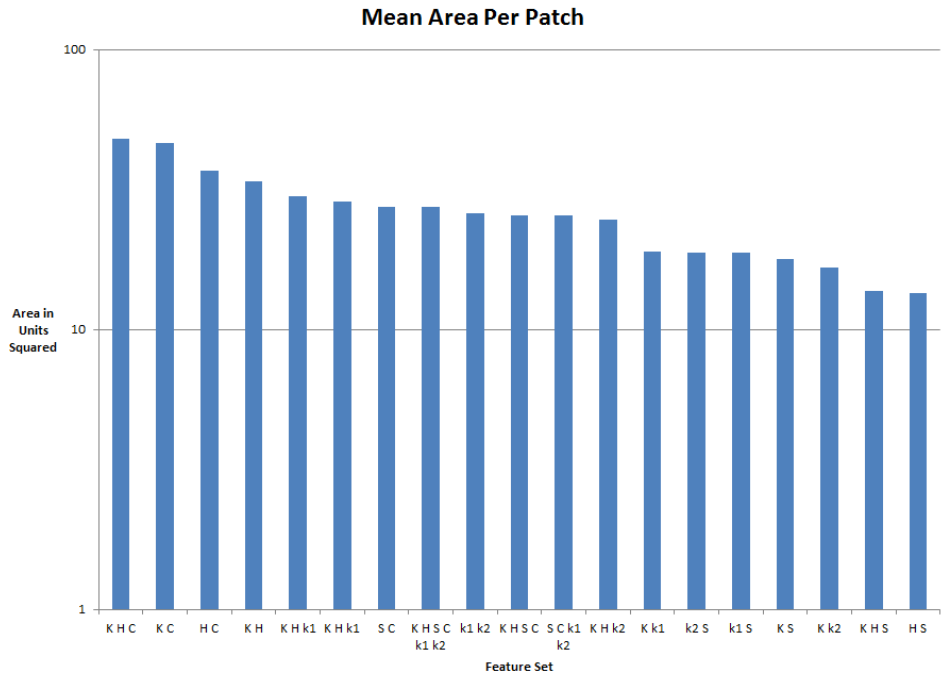
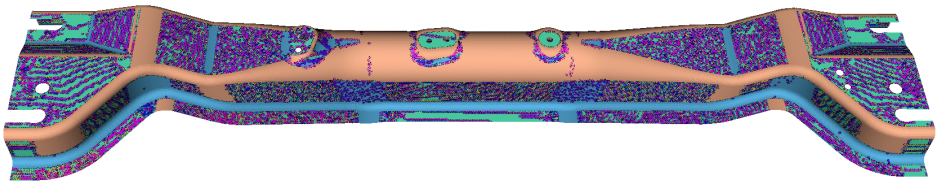


Figure 4-38: Random surface data set: Mean Patch area when varying SOM features

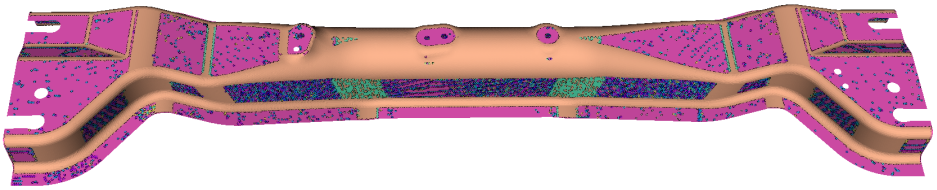
patch size and observed similar problems handling the planar region, but to a lesser degree compared to the HS feature set.

To see if a reduced SOM size would help improve the handling of planar regions we trained a 3×3 SOM using the feature set KHC (Figure 4-40) which produced the largest mean patch size. We found that though there was an improvement, the core weakness of this training data set was not eliminated and there were still planar regions being broken into noisy small sections.

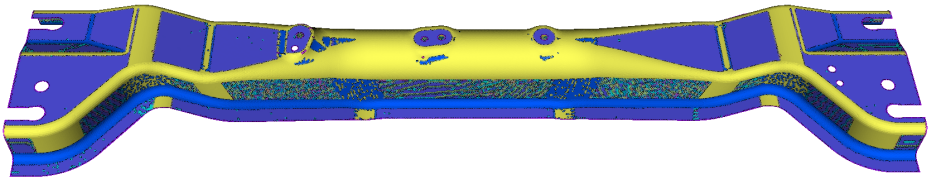
The feature set HC (Figure 4-42) produces the segmentation with the single largest patch (Figure 4-41) and suffers from the same problem as the other feature sets that segment planar regions into many small regions. When comparing the results of different features and training data sets the percentage of area covered by the largest patch is not as useful as the mean patch size when comparing results. This is because we are concerned with the overall segmentation, not an attribute associated with a single part of the segmenta-



(a) HS



(b) SC



(c) KHC

Figure 4-39: View of surfaces classified and segmented using a 4×4 SOM trained on random free-form surfaces

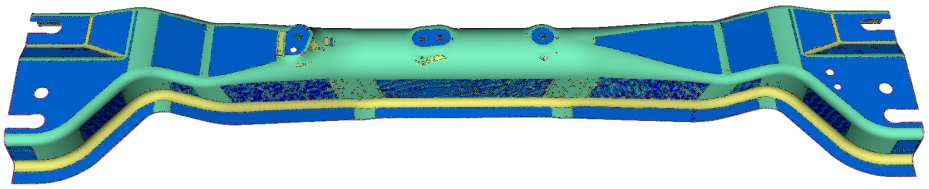


Figure 4-40: View of surface classified using a 3×3 SOM trained on random free-form surfaces using the features KHC

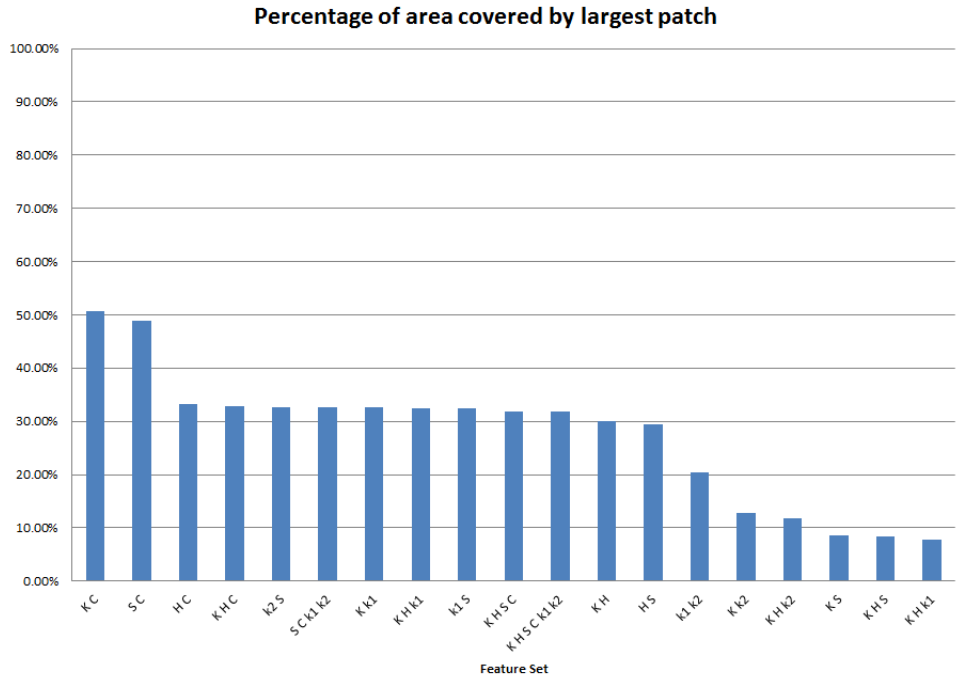


Figure 4-41: Random surface data set: Percentage of surface area covered by largest patch when varying SOM features

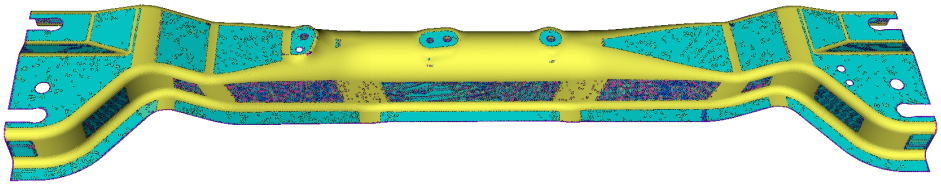


Figure 4-42: View of surface classified using a 4×4 SOM trained on random free-form surfaces using the features HC

tion.

When comparing the percentage of neurons fired (Figure 4-43) when varying features for the random surface training data set with the percent fired for the paraboloid data set (Figure 4-18) we observed a difference between the two sets with the random surface data set hitting 100 percent for half of the feature combinations and overall a higher percentage fired compared to the paraboloid data set. Considering the high level of noise on the planar regions for the random surface training data set compared to the paraboloid data set, a high percentage coverage of neurons isn't necessarily an indicator of a good training data set and may be an indicator of over-segmentation.

Comparing the resultant segmentation for the feature sets that generate the lowest and highest mean Q error (Figures 4-44 and 4-45), we find that the feature combination with the lowest Q error is able to segment curved regions with greater sensitivity. The dark blue regions at either end of the CAD model (Figure 4-45(b)) and the pink regions around the centre of the model are examples of this. It misses the curved sections between the planar regions on the top of the model marked in yellow for the feature set $KHSCk_1k_2$ shown in Figure 4-45(a). Based upon these observations the Q error metric is not necessarily an indicator of a good feature set selection.

Based upon our observations from inspecting the results of the segmentation when varying feature sets for the random surface data set using a 4×4 SOM, the best segmentation is generated from the KHC data set. We also found that the best indicator so far of the

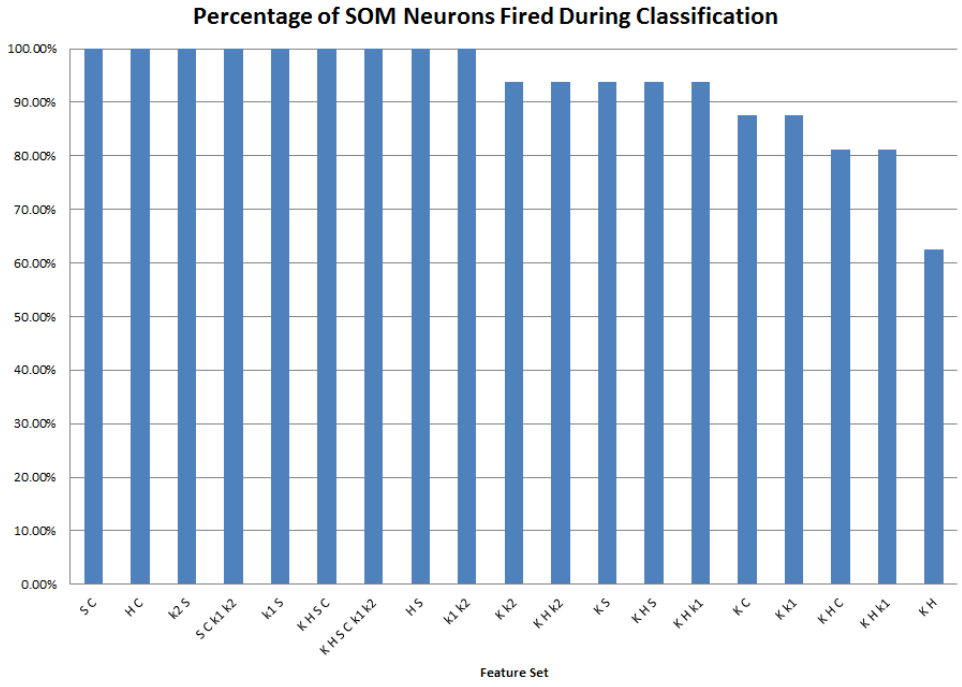


Figure 4-43: Random surface data set: Percentage of neurons fired during classification when varying SOM dimensions

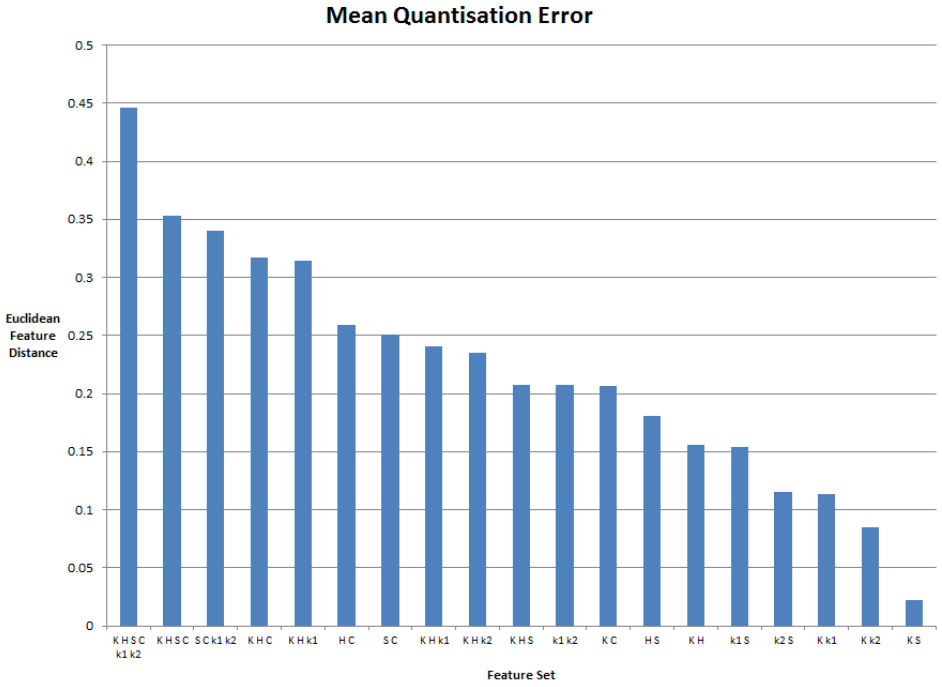
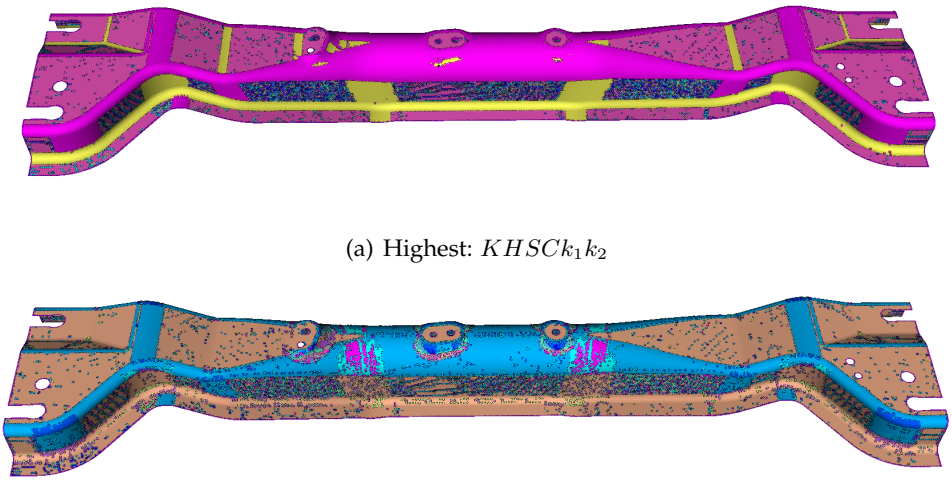


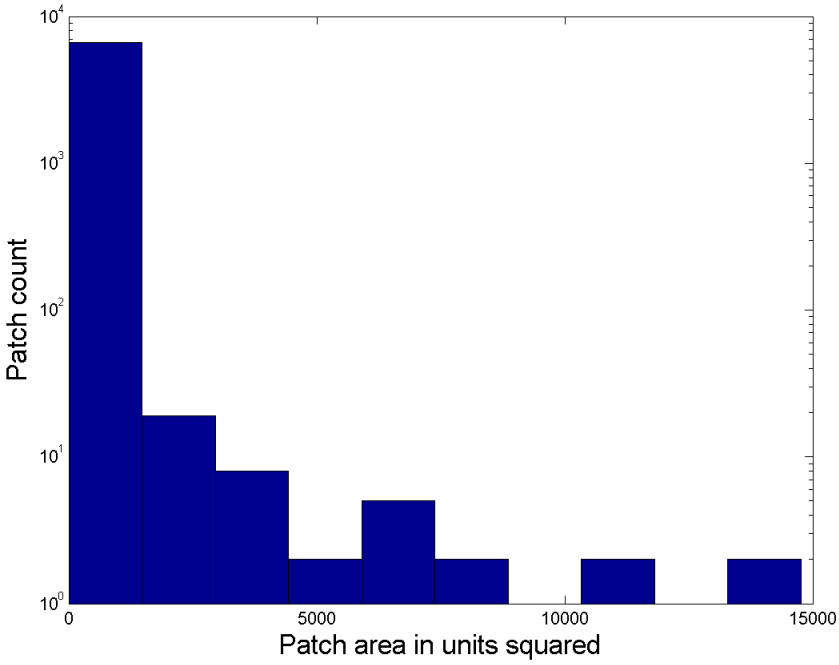
Figure 4-44: Random surface data set: Mean Quantisation error when varying SOM features



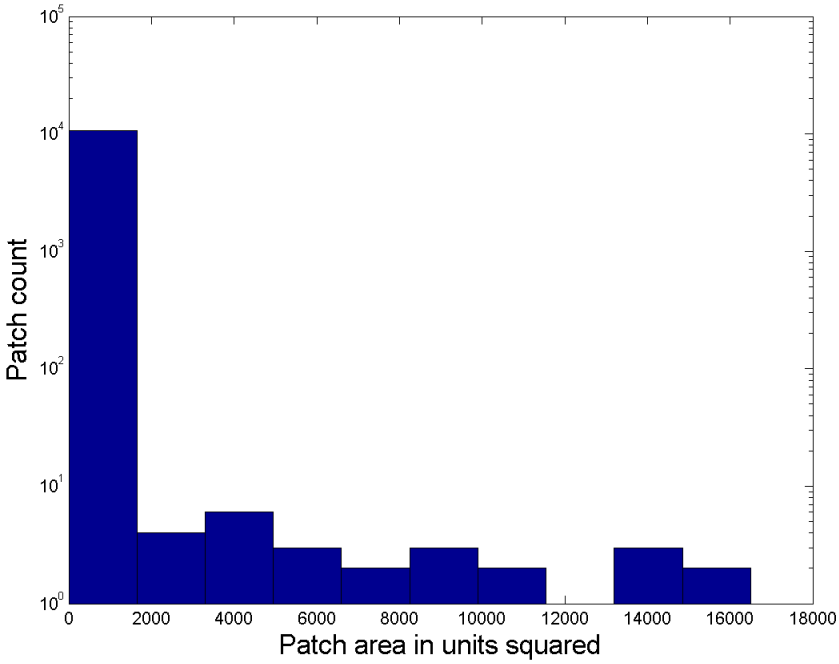
(a) Highest: $KHSCk_1k_2$

(b) Lowest: KS

Figure 4-45: View of surfaces classified and segmented using a 4×4 SOM trained on random free-form surfaces that have the highest and lowest mean Q error



(a) Highest: $KHSCk_1k_2$



(b) Lowest: KS

Figure 4-46: Patch size distribution with a log scaled y axis for the results in Figure 4-45

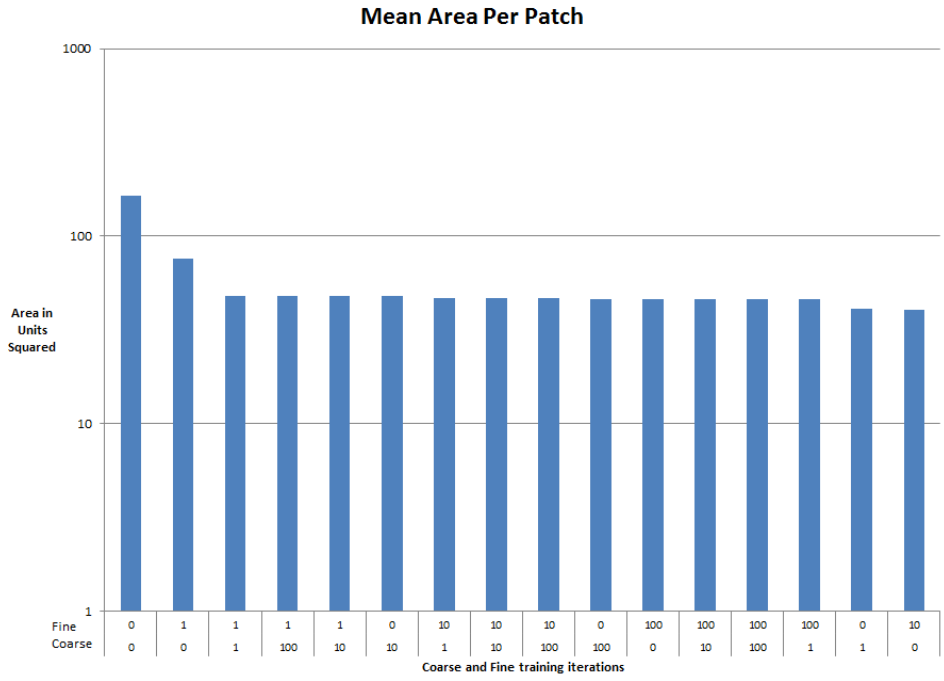


Figure 4-47: Random surface data set: Mean area per patch KHC iteration variation

behaviour of a SOM when segmenting the CAD model is to use the mean patch area when comparing results. We will use this idea combined with a binary search approach to find the best result when varying the training iterations.

4.2.3 Training Iterations

Using the feature set KHC and a 4×4 SOM we varied the training iterations to see what affect it has on the segmentation of the CAD model. The mean area per patch (Figure 4-47) is very stable compared to the results for the paraboloid training data set (Figure 4-24.) In both training data sets the degenerate case where there are 0 fine and 0 coarse training epochs, and n coarse and 0 fine epochs where n is the number of features in the training data set, produced the highest mean area per patch.

Both segmentation results at either end of the mean patch size plot, ignoring the degen-

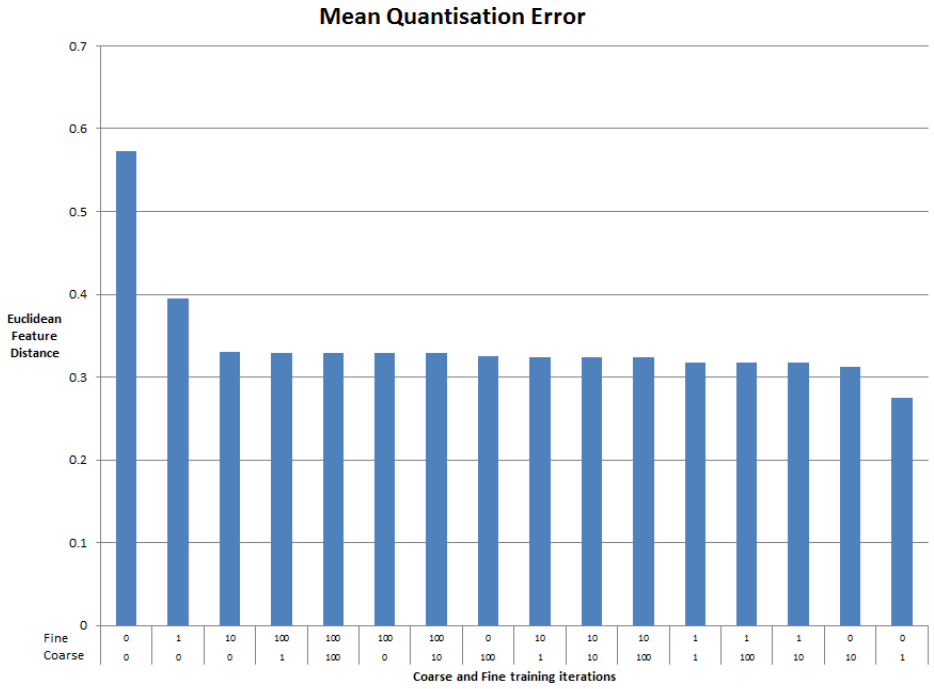
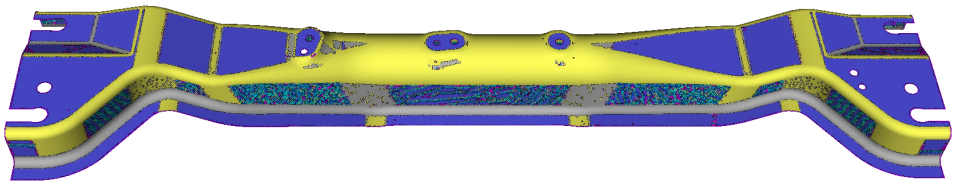


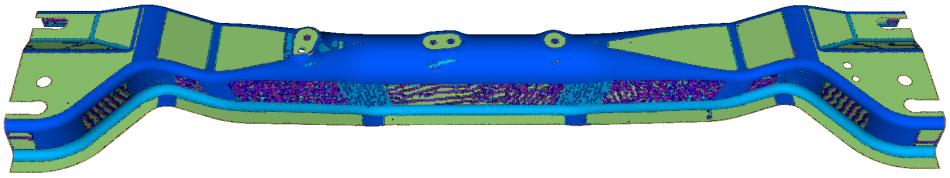
Figure 4-48: Random surface data set: Mean Quantisation error when varying SOM training iterations using KHC

erate case of 0 fine and 0 coarse epochs, produce almost identical segmentation results (Figure 4-49). The mean Q errors plotted in Figure 4-48 show the same epoch combinations occupying the top two entries in both plots with the rest clustered around a similar value. For the same feature set KHC using the paraboloid data set, the two identical epoch counts occupy the top positions for mean area per patch (Figure 4-24). The output of both SOMs in Figure 4-49 illustrates that varying the training iterations does not improve the poor segmentation results for the near-planar regions on the sides of the CAD model. The patch size distributions for these SOMs (Figure 4-46) are similar in patch counts and distributions.

The mean Q error for the random surface data set for the features KHC is greater than that of the paraboloid data set (Figure 4-51) by an order of magnitude. The resultant segmentation using the random surface data was better compared to the paraboloid data set when using the feature set KHC, indicating that very low Q error isn't always desirable



(a) 1 Coarse, 0 Fine training epochs



(b) 10 Coarse, 0 Fine training epochs

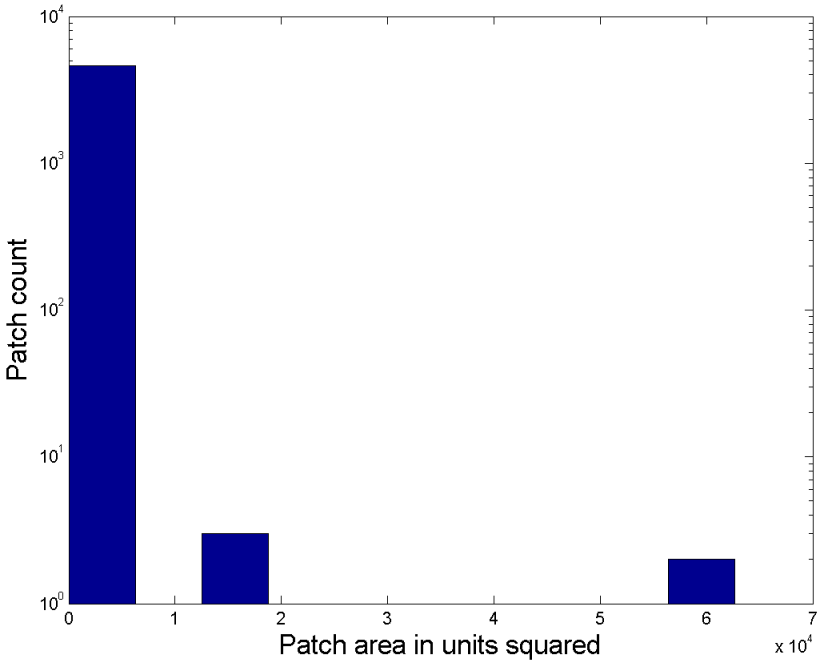
Figure 4-49: View of surfaces classified and segmented using a 4×4 SOM with KHC trained on random free-form surfaces

for good segmentation.

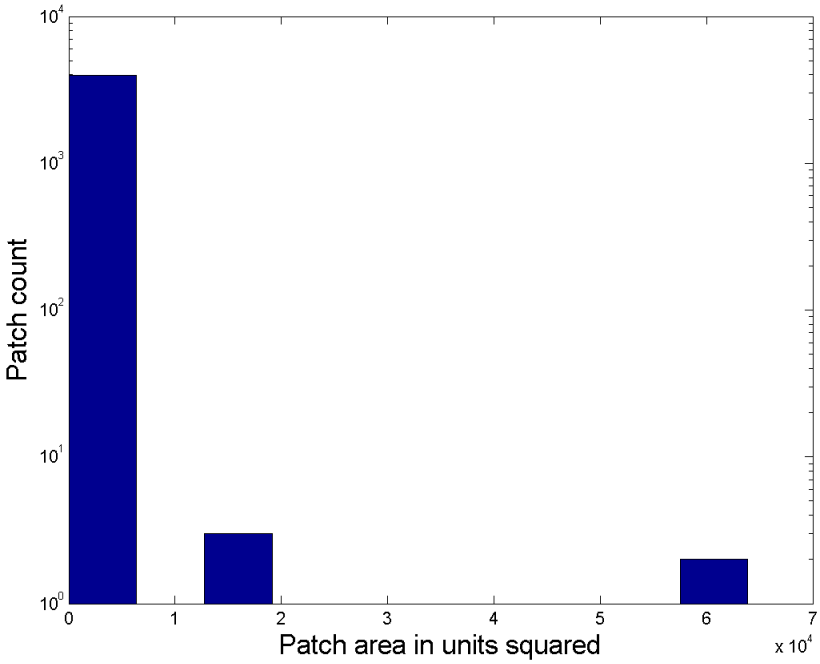
A consistent observation when comparing the paraboloid and random surface data set are the good results achieved using the feature set KHC. We will now investigate the performance of this feature set and the relationship between mean Q error and how well the SOM segments the CAD model.

4.3 CAD Model

Using the concept of self-similarity in a training data set we will train a SOM using the object that is being segmented. By definition the distribution of features within the training data set will be a good representation of the object.



(a) 1 Coarse, 0 Fine training epochs



(b) 10 Coarse, 0 Fine training epochs

Figure 4-50: Patch size distributions with a log scaled y axis for the surfaces in Figure 4-49

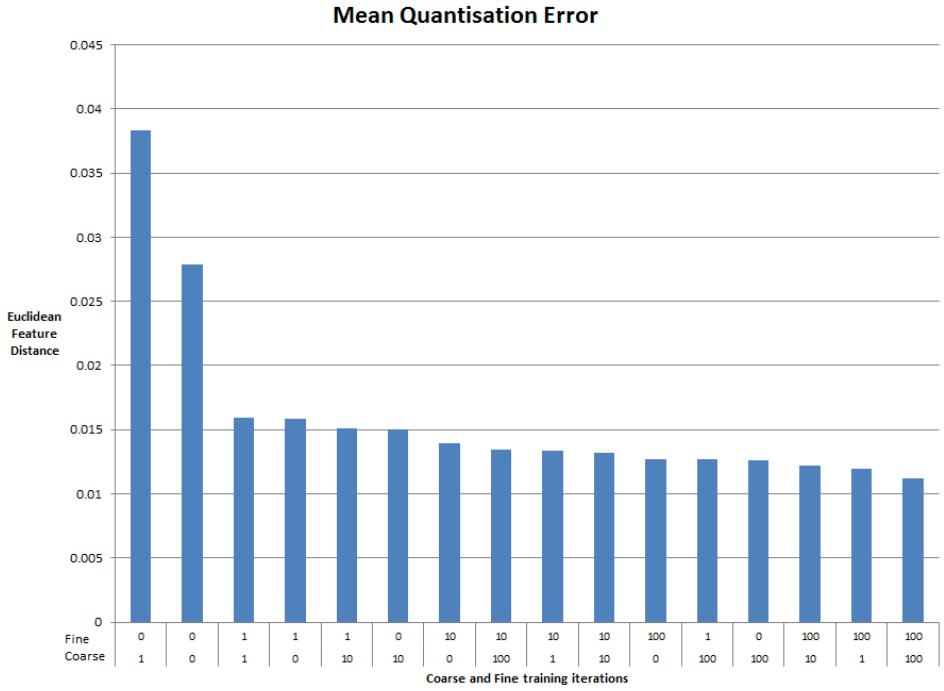


Figure 4-51: Paraboloid data set: Mean Quantisation error when varying SOM training iterations using KHC

4.3.1 SOM Size

The mean Q error for the feature set KH (Figure 4-52) using the CAD model itself as the training data set is on average several orders of magnitude less than that of the Random Surface and Paraboloid data sets for the same feature set when varying the SOM size. This is an indicator of how well the SOM clusters match the clusters of the CAD model. However, based upon previous observations this is not necessarily an indicator of good segmentation.

In the same way the mean Q error stabilised rapidly so did the mean area per patch (Figure 4-53). We didn't plot the patch distributions as the earlier sections demonstrated that the mean patch size plot and patch size distribution conveyed a similar message. The percentage of neurons fired during classification (Figure 4-54) remained high, compared to the paraboloid and random surface training data sets, which is an indicator that most,

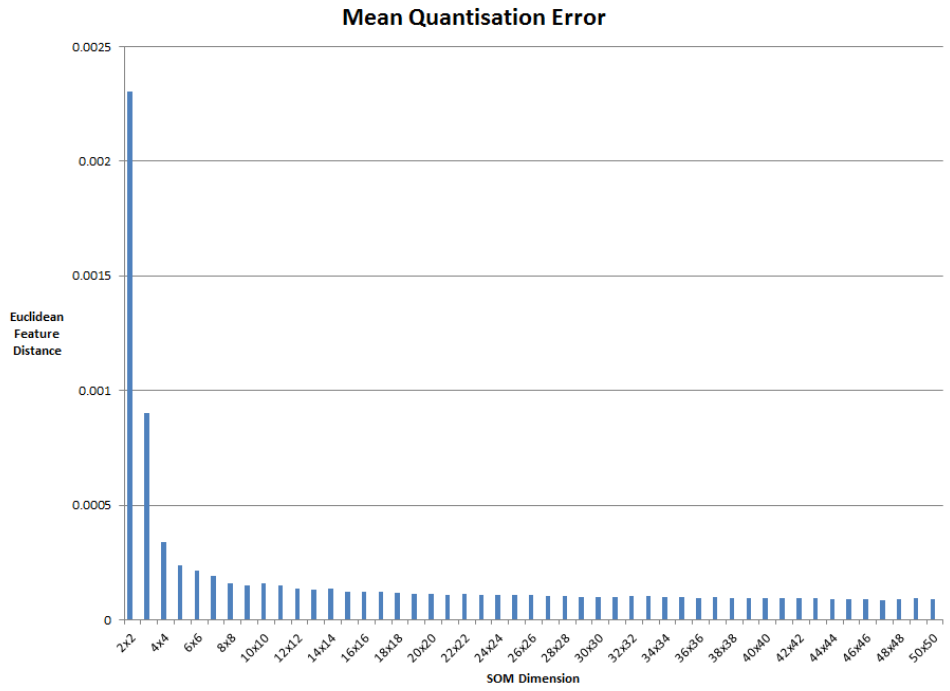


Figure 4-52: CAD model data set: Mean Quantisation error when varying SOM dimensions

and in some cases all, of the neurons of the SOM are being used to cluster features of the CAD model. The behaviour of the SOM as the size changes is also predictable for the feature set KH.

The predictability and stability of the SOM properties as it increases in size also applies to the segmentation output shown in Figure 4-55. As the SOM increases in size the number of segmented regions also increases as the SOM is able to describe more clusters. The initial 2×2 SOM in Figure 4-55(a) isn't able to perform any segmentation. This may be due to too much feature variation during the training process preventing the SOM from converging on a meaningful solution.

When the SOM increases in size the number of segmented regions and level of detail increases. The 3×3 SOM in Figure 4-55(b) segments the CAD model into regions that are sharply concave (grey-purple), concave (pink), convex (purple) and mildly concave,

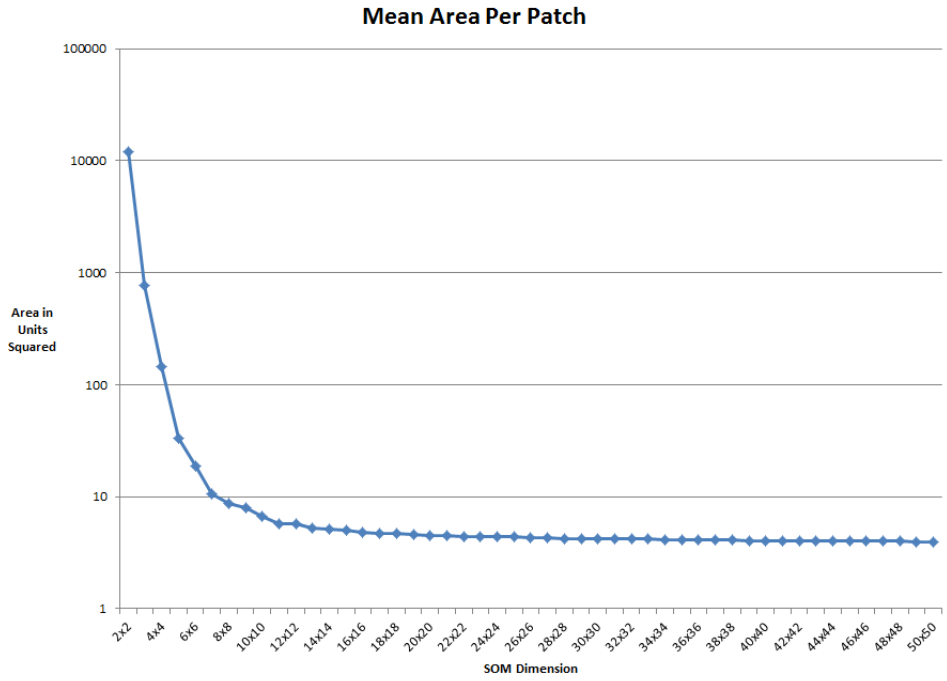


Figure 4-53: CAD model data set: Mean Patch area when varying SOM dimensions

convex or flat (blue). The transition to the 4×4 and 5×5 SOMs greatly increases the SOM's descriptive power and many more regions are extracted. These regions closely match the IGES model (Figure 4-2) that was converted into the triangulated mesh used in our investigations. When the SOM size increases smaller segmented regions are extracted from the model and are considered noise when sufficiently small as shown in the segmentation output of the 8×8 SOM in Figure 4-55(e). This coincides with the decrease in mean Q error, mean patch size and drop in the percentage of neurons fired shown in Figure 4-54. As the SOM size increases the mean Q error and mean area per patch are stabilising and the percentage of neurons fired steadily decreased.

4.3.2 Feature Selection

To evaluate how features impacted segmentation we used a 5×5 SOM which was the largest SOM that fired all neurons during classification from the last Section with the

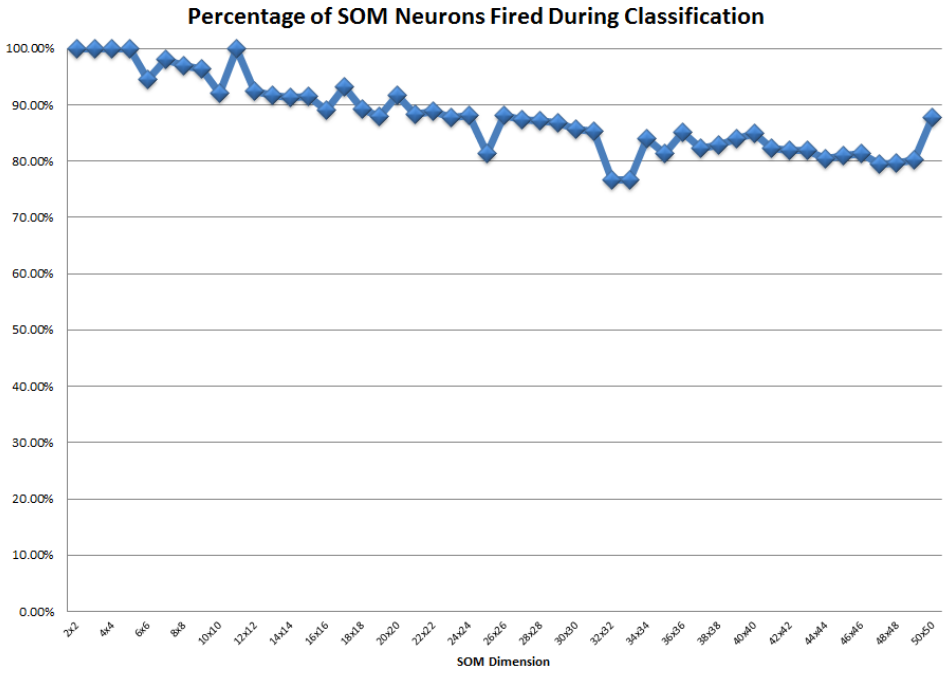
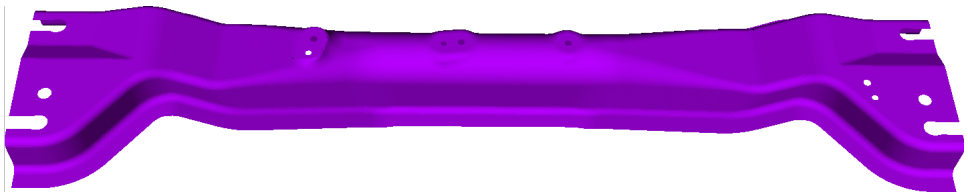
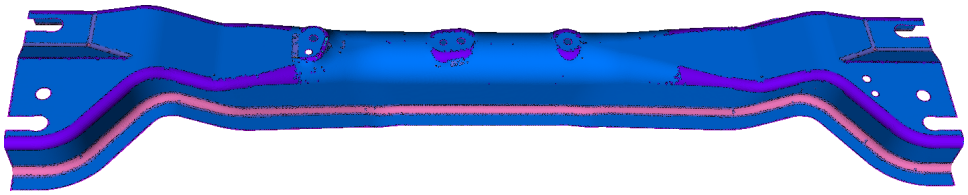


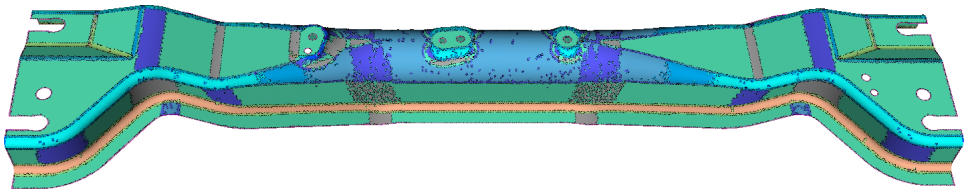
Figure 4-54: CAD model data set: Percentage of neurons fired during classification when varying SOM dimensions



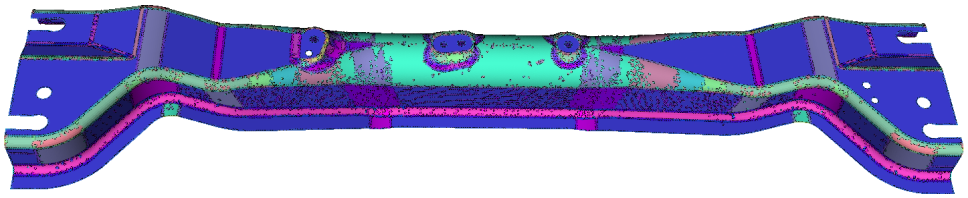
(a) 2×2



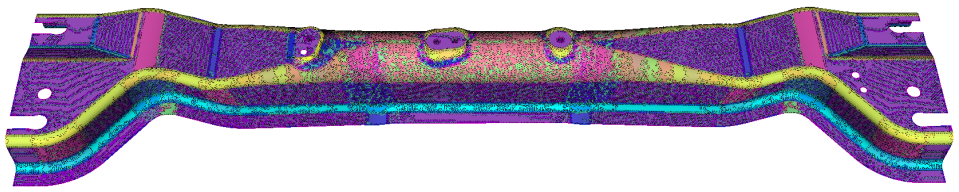
(b) 3×3



(c) 4×4



(d) 5×5



(e) 8×8

Figure 4-55: View of surfaces classified and segmented using a SOM trained on the surface being classified

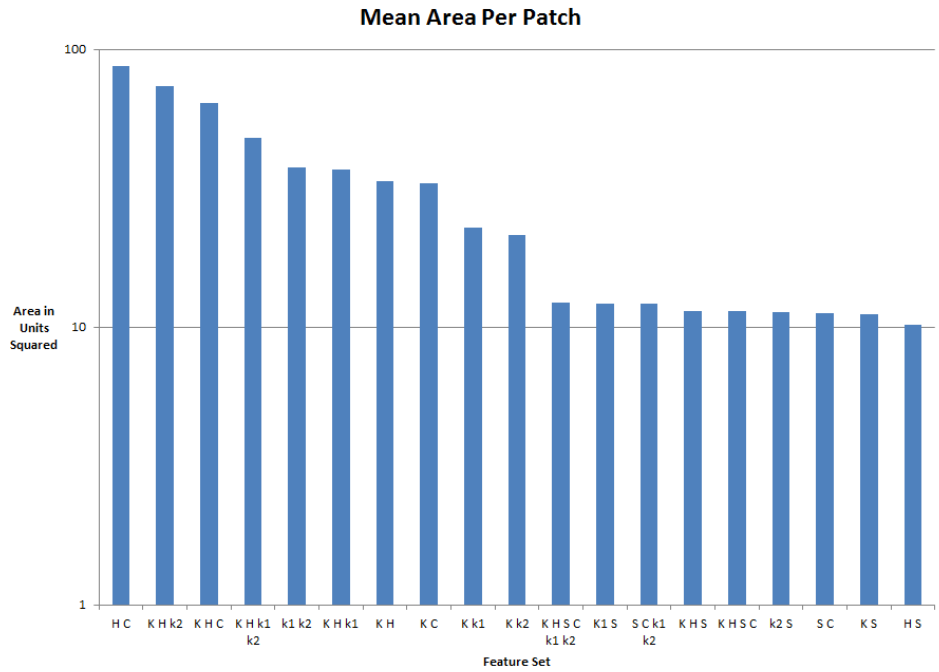


Figure 4-56: CAD model data set: Mean Patch area when varying SOM features

features KH. We did observe some noise on the planar to semi-planar regions on the side of the CAD model (Figure 4-55(d)) and we will use this characteristic of the KH feature set as a part of the comparison between feature combinations.

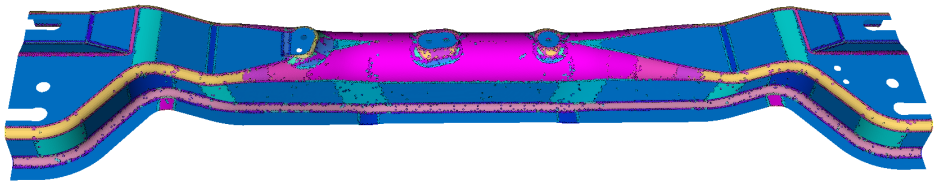
The mean area per patch (Figure 4-56) for each of the feature variations falls into two buckets, the features on the left hand side of the bar chart with a mean area per patch greater than 10 units squared and the features on the right hand side with smaller mean patch areas. The feature combination HC (Figure 4-57(a)) has the largest mean area per patch and produces distinct regions where the general curvature across all vertices in each region are bending in the same way. Examples of this include the blue planar regions, light blue convex regions with curvature in one direction, dark blue concave regions with curvature in one direction and yellow convex regions with curvature dominated in one direction. There are some subtle bends that are not picked up by the classifier such as the yellow region, but on the whole this is a reasonably clean segmentation.

Images of a selection of the segmented CAD part with different feature combinations in order of mean patch size in Figure 4-57 illustrates the relationship between mean patch size and segmented output. The larger segmented regions are similar with the noise on the near-planar surfaces affecting the output. This noise spreads to the curved regions as reflected in the mean patch size measurement.

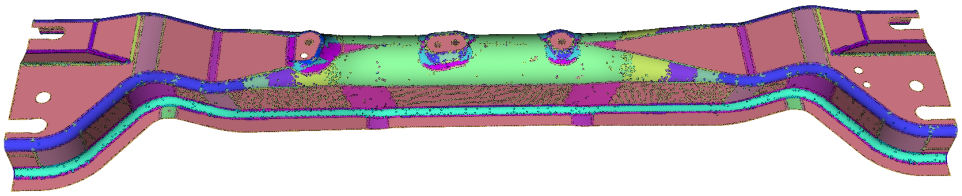
The cleaner segmentation results are dominated by feature combinations that contain H, C, K and this result is similar for the Paraboloid and Random Surface training data sets in Figures 4-14, 4-37, 4-39(a), 4-40 and 4-42.

The percentage of neurons fired during classification for the CAD model training data set (Figure 4-58) is higher on average than for the Random Surface (Figure 4-18) and the Paraboloid (Figure 4-18) data sets. The segmentation results of the CAD and Random Surface data sets are far better than what was achieved for the paraboloid data. Varying shades of pink for both planar and curved regions, using both a colour to indicate cluster membership and black lines to indicate boundaries (Figure 4-15) suggest that the only reason there was a good segmentation was the uniform initialization and large number of neurons rather than any special properties of the Paraboloid data set.

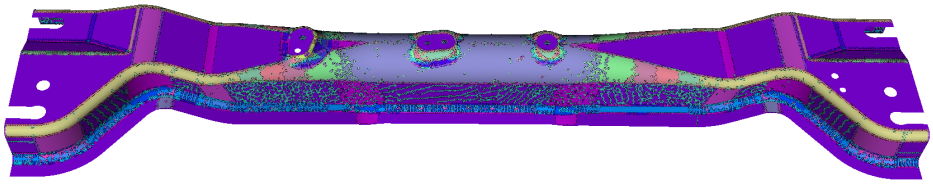
When comparing the mean Q error of the feature combinations between each of the training data sets, the CAD dataset produces a maximum Q error (Figure 4-59) that is two orders of magnitude less than that of the Paraboloid data set (Figure 4-19) and an order of magnitude less than for the Random Surface data set (Figure 4-19). This relatively low Q error was achieved with a 5×5 SOM which is slightly larger than the 4×4 SOM used with the random surface dataset and far smaller than the 15×15 SOM used with the Paraboloid data set. The larger SOM used for the Paraboloid data to achieve poorer results is more evidence that the Paraboloid data set is not suited to segmenting the CAD model and the only way it segmented the CAD model was by virtue of its size.



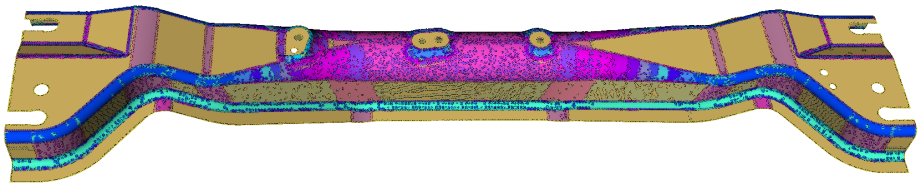
(a) HC



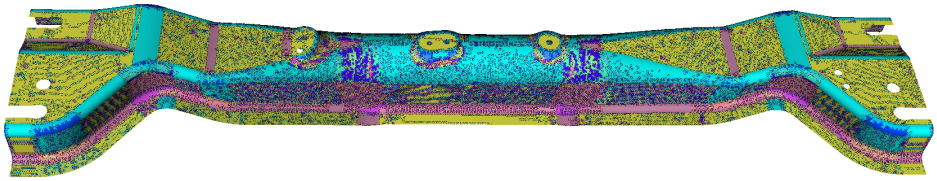
(b) KHC



(c) k_1k_2



(d) KC



(e) HS

Figure 4-57: CAD model data set: Feature variation ranked by mean patch size descending

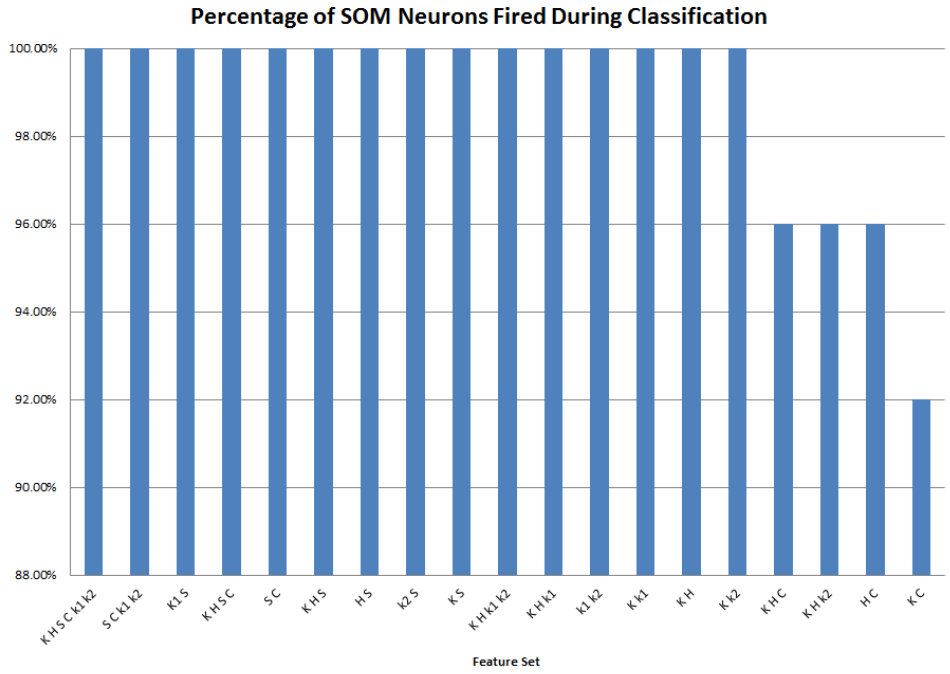


Figure 4-58: CAD model data set: Percentage of neurons fired during classification when varying SOM features

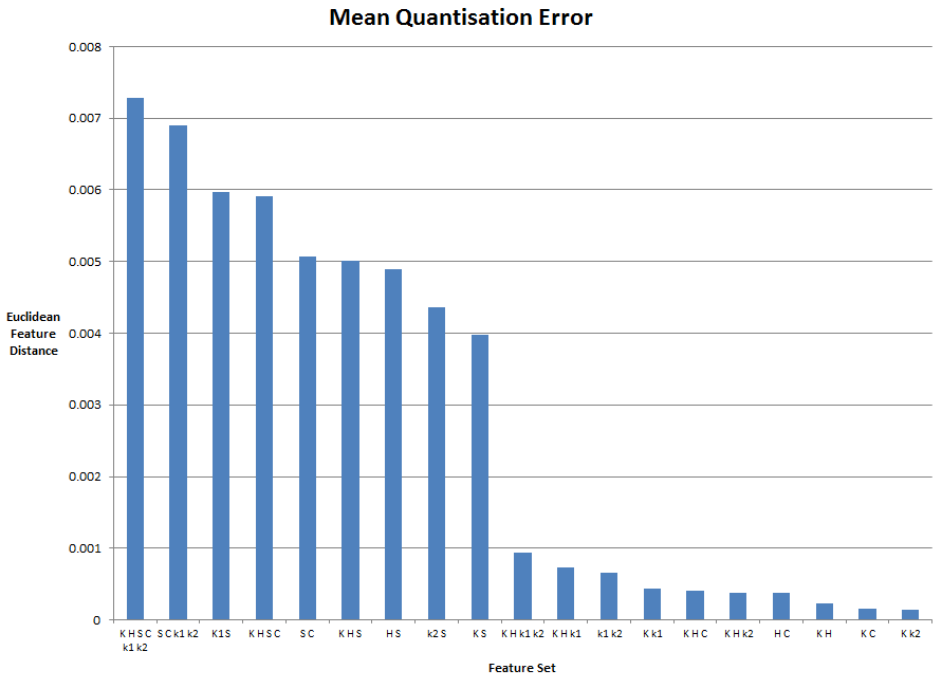


Figure 4-59: CAD model data set: Mean Quantisation error when varying SOM features

4.3.3 Training Iterations

We discarded the results from using 0 fine and 0 coarse training epochs as earlier results showed the resultant SOM degenerated into a bucketing function and performed poorly. There were two distinct groups when looking at both mean Q error (Figure 4-60) and mean area per patch (Figure 4-61). Those with fine training iterations and those without. For both sets of results the mean Q error and mean area per patch are similar.

The results of segmenting the CAD model using 0 and 100 fine training epochs and the same coarse training epoch count shown in Figure 4-62 are very similar. For example the large curved region at the centre of the model marked in purple in Figure 4-63(a) and yellow in Figure 4-63(b) are the same. The only noticeable difference between the two models is how the very slightly curved regions that are at the centre of the CAD model in Figure 4-63, which are light yellow for 0 training epochs and dark blue for the 100 fine training epochs. Here we can see the fine training iterations make the model more

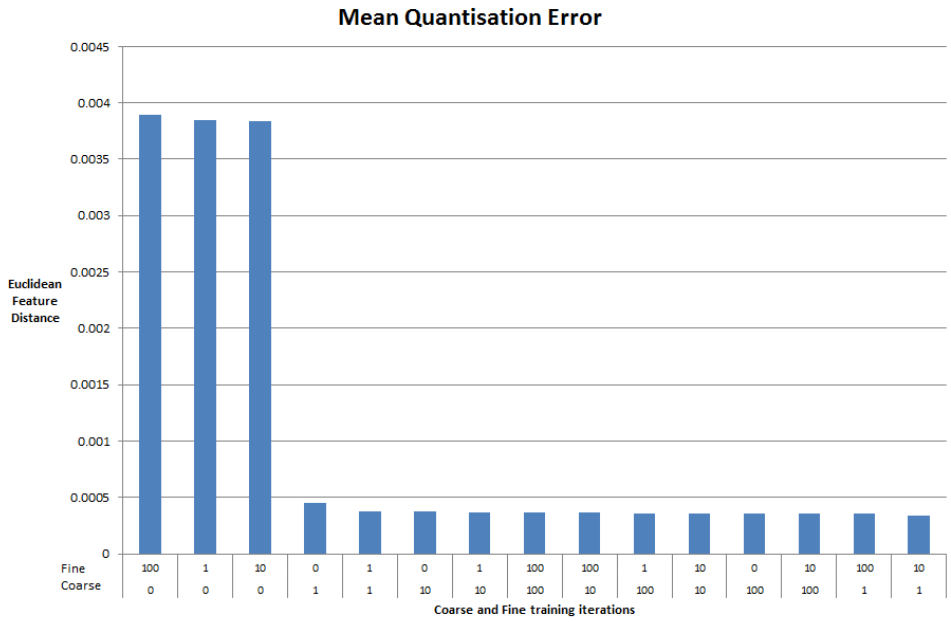


Figure 4-60: CAD data set: Mean Quantisation error when varying SOM training iterations using HC

sensitive to slight changes in curvature which appear to be imperceptible and could be artefacts of the meshing of the surface and not an intrinsic property of the CAD model.

The results shown in Figure 4-63 raises the question of whether fine training iterations are required. The coarse training iterations used in the previous comparison was the maximum for our experiment so we reduced it to 10 to avoid possible over sensitivity of the SOM introduced during the training process. We compared the results of 0, 1, 10 and 100 fine training epochs. The segmentation results (Figure 4-64) shows that when setting the fine training epochs to 0 (Figure 4-64(a)) there is much noise in the segmentation that drops significantly when fine training epochs are set to 1 and then increases as the number of fine iterations increases beyond that value. This illustrates the balance that needs to be found when determining if the SOM has been overtrained, isn't sensitive enough or is just right for the purposes of segmentation.

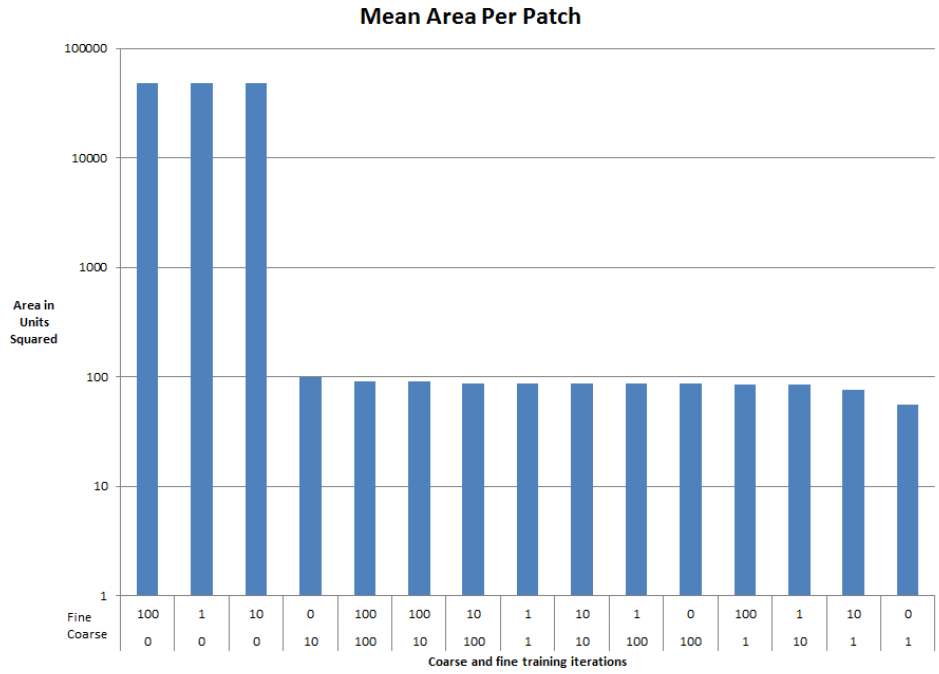
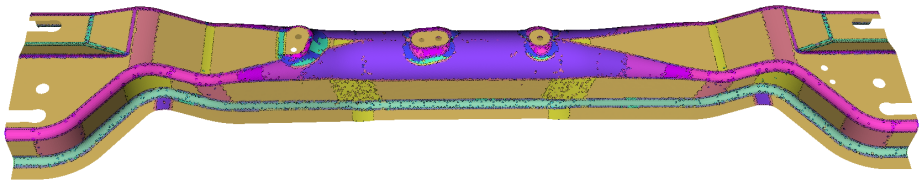
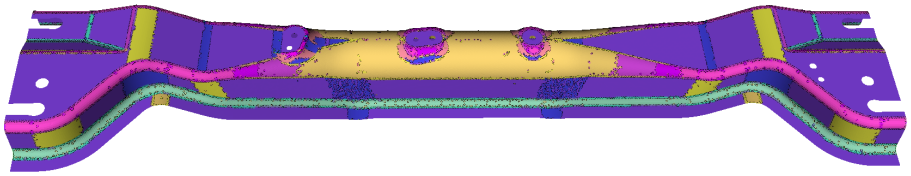


Figure 4-61: CAD data set: Mean area per patch HC when varying SOM iteration variation

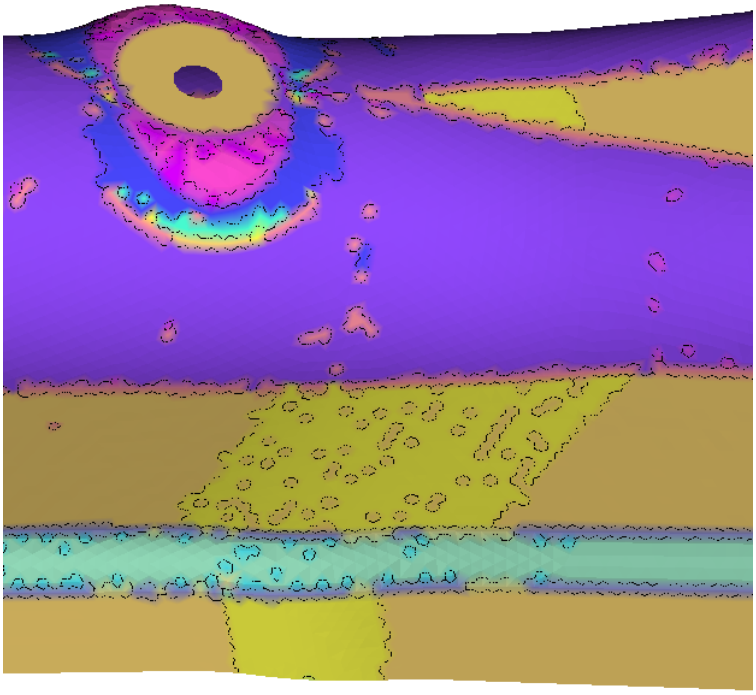


(a) 100 coarse, 0 fine

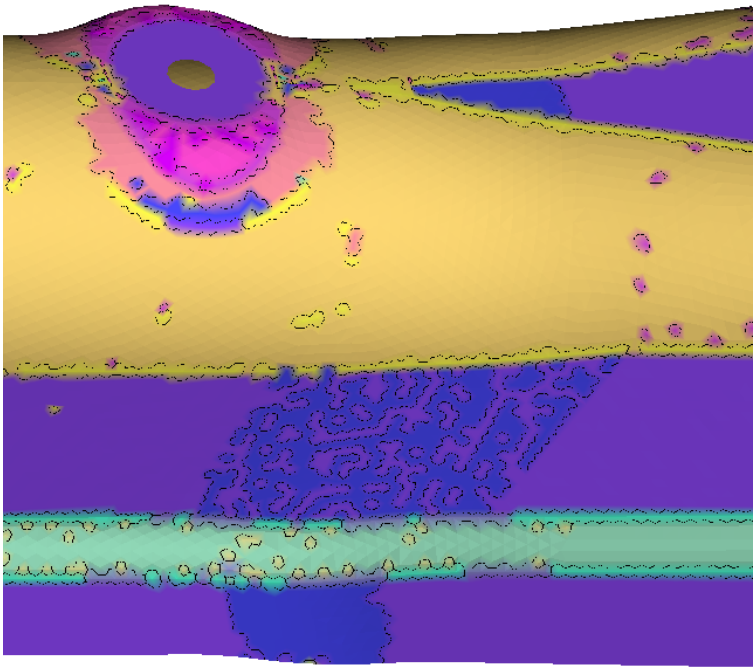


(b) 100 coarse, 100 fine

Figure 4-62: Comparison of the segmentation results for 100 and 0 fine training epochs.

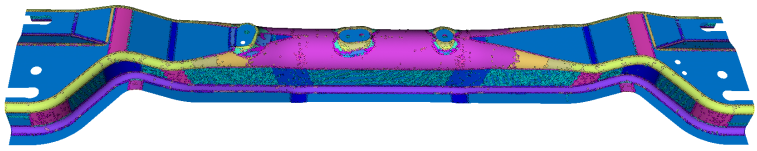


(a) 100 coarse, 0 fine

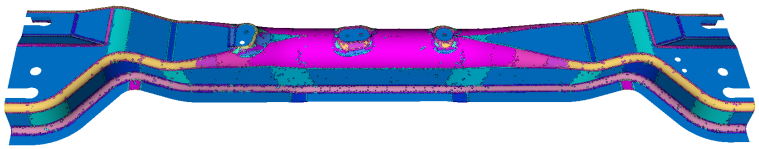


(b) 100 coarse, 100 fine

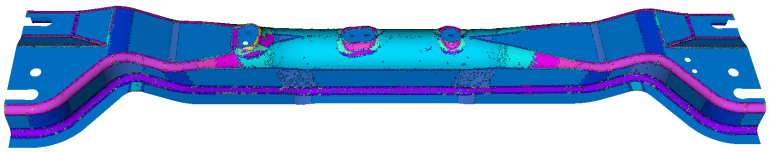
Figure 4-63: Detail of comparison of the segmentation results for 100 and 0 fine training epochs.



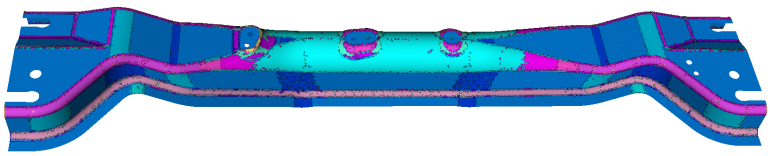
(a) 1 coarse, 0 fine



(b) 1 coarse, 1 fine

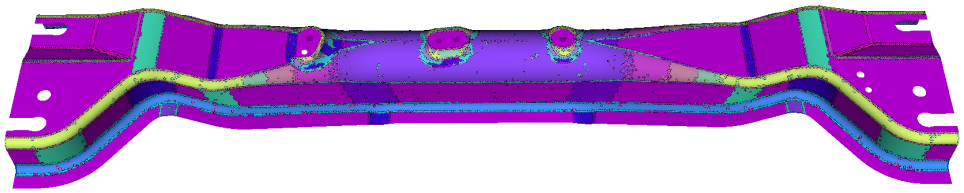


(c) 1 coarse, 10 fine

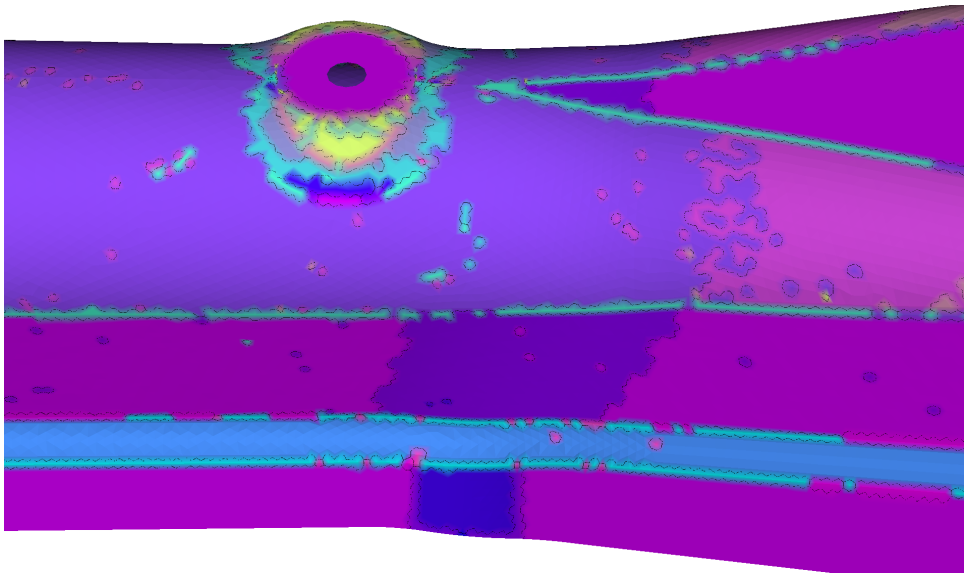


(d) 1 coarse, 100 fine

Figure 4-64: Comparison of the segmentation results for 10 coarse training epochs while and varying fine training epochs.



(a) Segmented Model



(b) Detail

Figure 4-65: Segmentation result for 10 coarse and 0 fine training epochs.

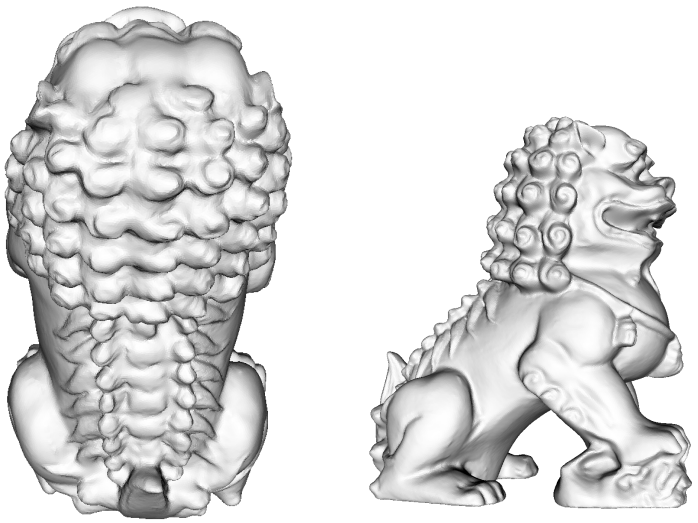
Given the good results for 1 coarse and fine training epoch we examined the result of training the SOM using 10 coarse and 0 fine training epochs to see if by simply increasing the coarse training iterations with no fine training we could end up with a SOM with a good balance of sensitivity. This combination, shown in Figure 4-65, gave us our best segmentation result for this Section and was able to perform well with slightly curved regions with little noise in the segmented output (Figure 4-65(b)). Note the similar curvature properties of segmented regions that share the same colour such as the pink planar regions, blue convex regions that are curved in one direction, and the green concave regions that are curved in one direction. One weakness of the features used in this investigation is illustrated by the yellow region where there are changes in the secondary curvature direction that are not picked up by any of the features used which suggests that an improvement could be made by introducing a feature that can detect such changes.

To demonstrate our technique we segmented additional 3D models using the same parameters from the previous Section, a 5×5 SOM using HC as features and 10 training epochs and training the SOM on the object being segmented. The hand crafted Fu-Lion acquired using a laser scanner shown in Figure 4-66 consists of many regions of complex curvature changes. We were able to segment this into regions (Figure 4-68) that aligned with the curvature variations observed by the eye, the model coloured by cluster membership is shown in Figure 4-67. This model proved to be very complicated in terms of variance in curvature and subsequently we couldn't extract large regions of significance but we could identify patterns of curvature types in the complicated sections such as the knows in the mane of the lion. Upon visual inspection we could see the large regions of consistent curvature on the flanks and chest of the model were extracted but the model is too complicated for us to draw too many conclusions about our technique from it. The segmentation was reasonably consistent as the colouring in regions of similar curvature types was similar.

The results of the segmentation process for each of the CAD models depended upon how varied the curvature was on the model. When using the same segmentation technique on the CAD models in Figure 4-69 the results varied greatly as the distribution of curvature types affected how well the training process identified the different combinations of curvature metrics. For objects that appeared to be complex the sheer number of vertices that showed very little curvature due to meshing parameter selection and the shape of the object caused the SOM to group regions that were visually different together in the same cluster, an example of this is shown in Figure 4-69(b). To deal with objects of this type we will investigate the use of exemplar surfaces in Section 5.2.

4.4 Conclusion

In this Chapter we investigated how the selection of training data, features and training parameters affected the segmentation of a CAD model by classifying its vertices and

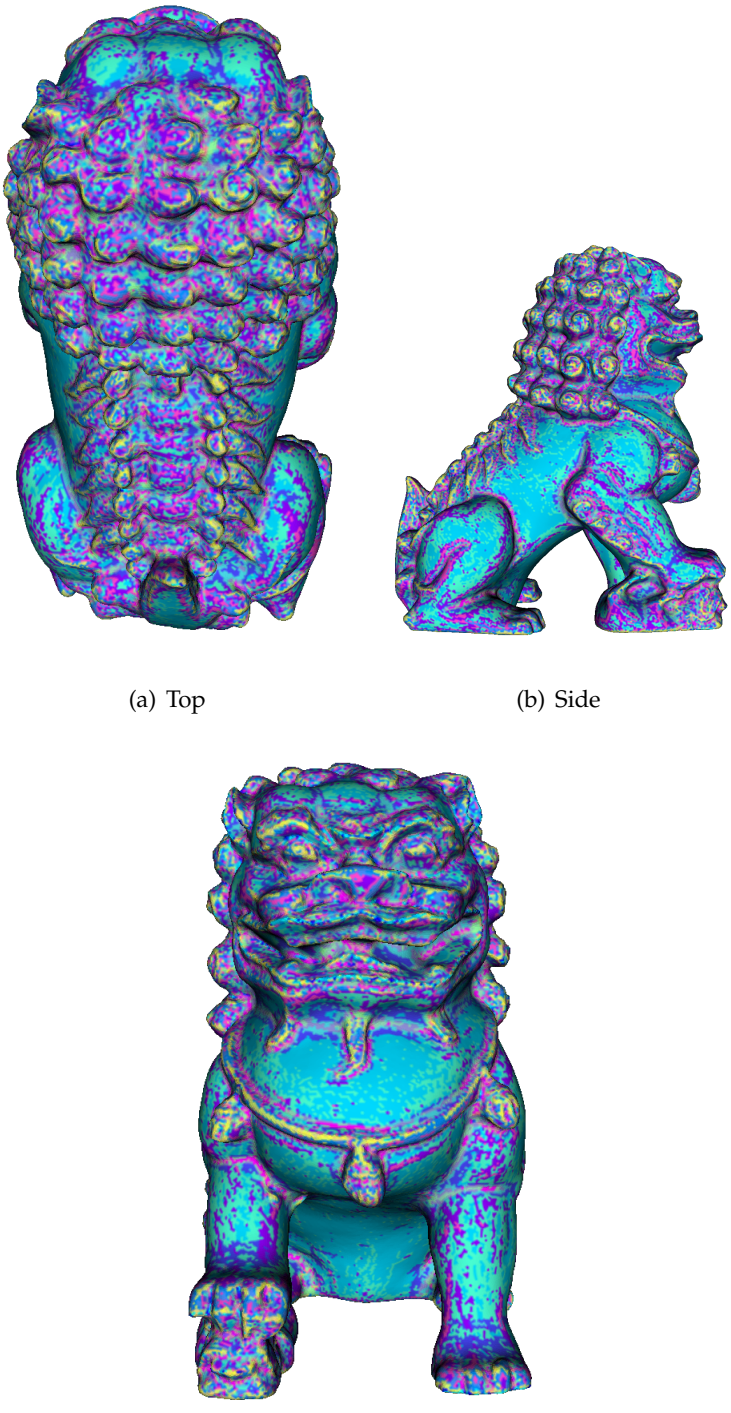


(a) Top

(b) Side

(c) Front

Figure 4-66: Rendered 3D model of Fu-Lion to be segmented [Falcidieno, 2009]

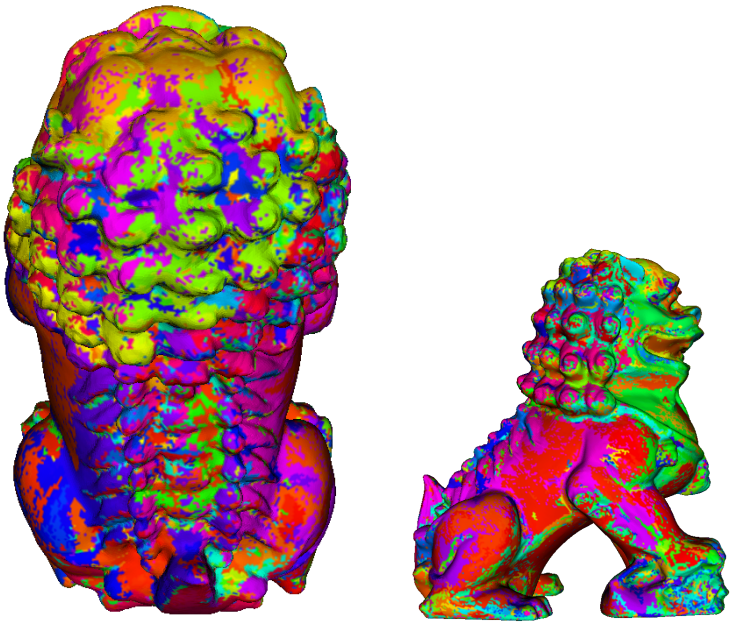


(a) Top

(b) Side

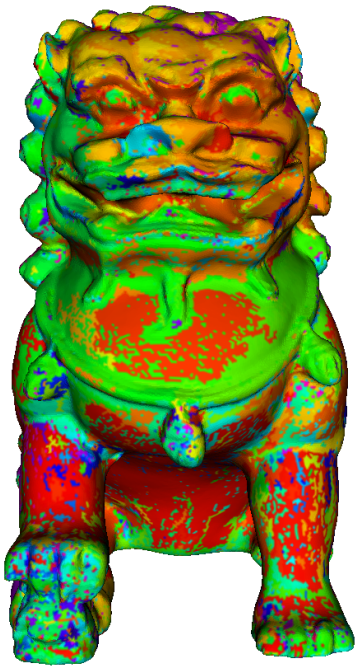
(c) Front

Figure 4-67: Classified Fu-Lion[Falcidieno, 2009]



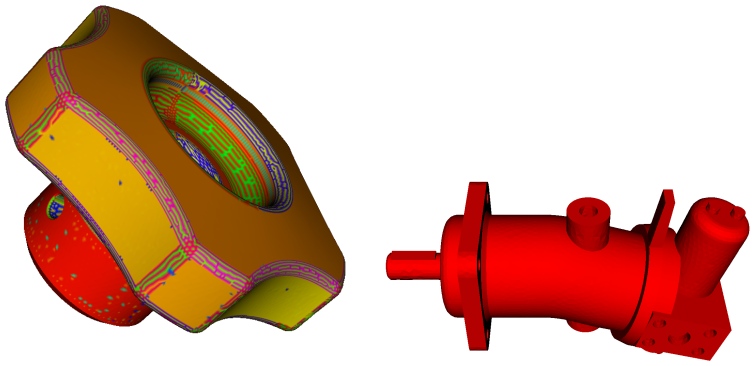
(a) Top

(b) Side



(c) Front

Figure 4-68: Segmented Fu-Lion[Falcidieno, 2009]



(a) (b)

Figure 4-69: Example of two surfaces segmented using itself as the training data

segmenting the CAD model by extracting regions of connected vertices that share the same BMU. Using the SOM as a bucketing function without training is not a reliable technique to segment objects as extreme feature values can cause the majority of vertices to be place in the same bucket. Once the SOM neurons have been trained they can better represent the distribution of features in the training data set.

We found that the features HC gave consistently good results across the different training data sets. This makes sense as both features use the mean value of k_1 and k_2 in different ways, balancing out possible extreme values of k_1 k_2, whereas K, which did perform reasonably well across the different training data sets, multiplies these values making any noise or large values significantly affect the result. Avoiding the boundary effect by using the toroidal topology hexagonal neighbourhood functions which gave us a different set of features that gave better segmentation results compared to MacLennan et al. [2006] which found KH to be the most effective feature combination.

The number of iterations used when training the SOM strongly affected the sensitivity of the SOM to stimuli. Too many fine or coarse training epochs added noise to the segmentation. The best result of this Chapter used no fine training epochs and 10 coarse training epochs to achieve a segmentation that closely resembled the regions in the underlying IGES model that was the basis of the triangulated mesh we were segmenting.

When segmenting models in the training data set the best SOM was the model being segmented. The SOM and segmentation characteristics such as mean Q error and mean patch size when using the CAD model as the training data set were consistent as the SOM size was varied for a fixed feature set. The paraboloid data set was unsuitable as a training data set as reasonable segmentation results appeared to be a product of SOM size and not of any properties of the training data set. This was illustrated by the small distribution of neurons fired during the classification process. The random surface dataset fared better than the paraboloids with a relatively high number of neurons being fired during the classification process, presumably because the random surface dataset had more in common with the CAD part than the paraboloids. The noise observed in the paraboloid data set when it was classified appeared to result in noisy segmentations in some regions of the CAD model. The best results were achieved using the CAD model training the SOM. In the next Chapter we will evaluate the effectiveness of these training parameters when using the SOM as a classifier for indexing and retrieving free-form surfaces.

Chapter 5

Indexing Methods

To facilitate the indexing and retrieval of free-form surfaces we need to reduce each surface into a set of features that can be simply expressed yet powerful enough to allow reasonably accurate matching. In Chapter 3 we described how the vertices of a mesh representation of a 3D model can be clustered using a Self-Organizing map and in Chapter 4 we used this technique to segment a free-form surface. We will use a classification technique to perform approximate region matching of 3D models by using the SOM response to a set of vertices from a spatially local region and representing that response as a histogram. This histogram representation will also be used to identify object classes. Our approach is analogous to two techniques: bag-of-words and shape distributions.

The bag-of-words approach was initially used in textual analysis where documents are classified by examining the frequency and presence of words without taking into account the grammatical structure of the content. This technique has also been used to train spam filters using a Bayesian classifier with a corpus of spam email content [M. Sahami and Horvitz, 1998]. Other learning techniques such as Latent Semantic Analysis and Latent Dirichlet Allocation use bag-of-words to identify topics, related documents and other properties of sets of documents [Dumais, 2004, Blei et al., 2002]. The same underlying concept of the bag-of-words approach has been successfully applied to image retrieval [Rubner et al., 2000, Wu et al., 2009] and object recognition [Botterill et al., 2009, Wu et al., 2009, Toldo et al., 2010, Gao et al., 2010, Zheng and Gao, 2008]. In each of these publications sets of features using the bag-of-words approach are used as the basis of indexing and retrieving objects and images.

Distributions of surface and volume based metrics have been used to perform both complete and partial object class similarity measurement. Osada et al. [2002] calculated distributions of distance, area and angle based measurements for random point pairs on the surface of a model and Vandeborre et al. [2002] combined a set of metrics based on volume, random point distance and shape index.

5.1 Regions represented as distributions

Reducing complex relationships to numeric distributions is a common technique for simplifying the task of identifying similarity. We will apply this technique to find regions on 3D models that share a common distribution of cluster membership weighted by the area around each vertex.

Using the classification technique described in Chapter 3 we classify the vertices of a 3D model and the surface, or part of the surface, and build histograms of neural responses to describe a region and discriminate it from others. The regions being compared could be whole surfaces or regions of a surface bound by a user selected section. This approach is analogous to the Bag-of-words approach of object recognition used in Dalal and Triggs [2005] and Gao et al. [2010] where regions are described using histograms of intensity gradients. No inter-feature spatial relations are encoded in the histograms that represent a bound region of an image. Our histogram representation is determined by calculating the proportion of the region that belongs to each cluster, that cluster being represented by a neuron that belongs to the SOM.

To demonstrate the histogram calculation process, we will take some simple shapes, train a SOM using them as input and use the same SOM to plot the resultant histograms from each shape. Using these plots and the distance calculation we can find which shapes share similar BMU distributions by using the sum of squared distances between each histogram.

Later in the Chapter we will use the Earth Movers Distance (EMD) metric to calculate the distance between histograms to find the best match. EMD [Peleg et al., 1989, Rubner et al., 1998] has been applied to problem domains such as content-based image retrieval [Tan and Ngo, 2009], 3D object retrieval [Gao et al., 2010], SIFT feature matching [Pele and Werman, 2009, 2008] and document similarity [Wan and Peng, 2005]. The metric is a work minimization function that calculates the cost of transforming one histogram of fixed intervals or signature (the bins are clusters) into another. The analogy used by

Figure 5-1: BMU to Colour Map for 3×3 SOM with the origin marked in the top left corner

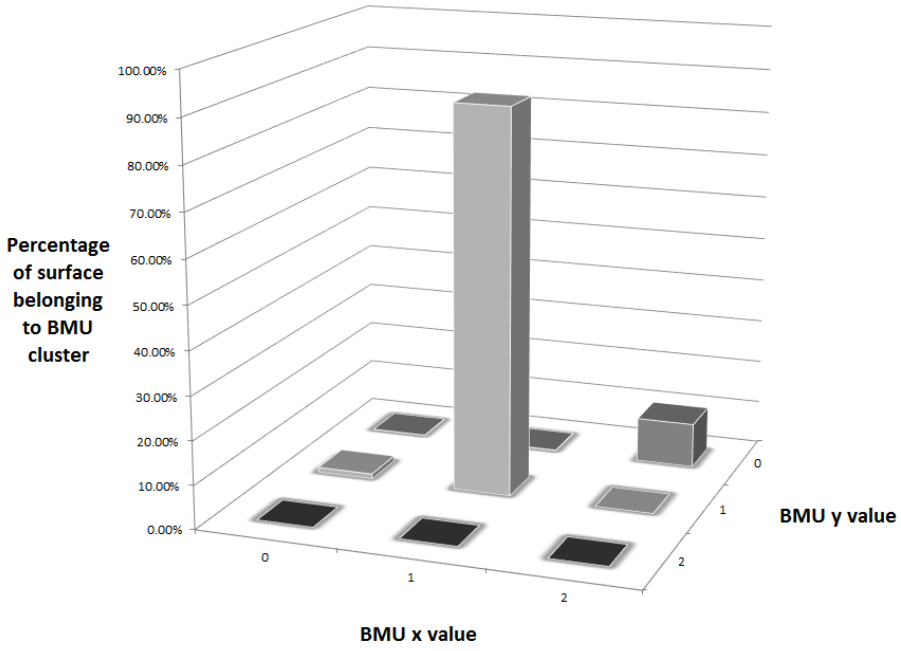
Rubner et al. [1998] to describe this function is the cost of moving mounds of dirt (source) into holes (target). They calculated this work by framing the problem as an optimal flow network with the work required for a source P, target Q, flow between two bins f_{ij} and distance between those bins d_{ij}:

$$EMD(P,Q) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} f_{ij}}{\sum_{i=1}^{d} m \sum_{j=1}^{n} f_{ij}}$$

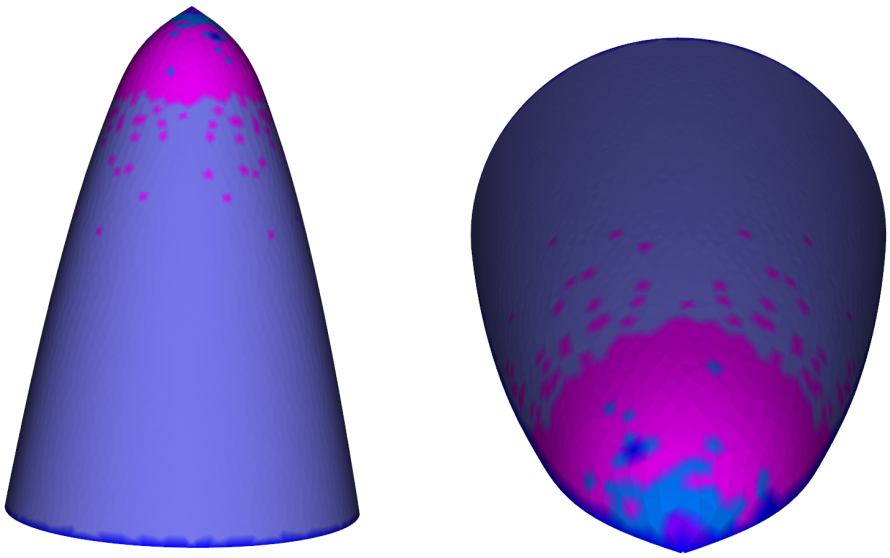
The EMD metric can account for the toroidal topology of the SOM as it uses d_{ij} to calculate distances between each of the buckets. We used a lookup table populated by the Euclidean distance between the neurons of the SOM making the EMD suitable for any shape or topology. We used the implementation provided by Pele and Werman [2009] via the Matlab to Java bridge JAMAL [Khadkevich, 2013] and implemented our own distance metric function by creating a lookup table of the Euclidean distance between each of the clusters in the SOM.

The shapes used in this example (Figure 3-1 on page 56) consists of a paraboloid, elliptic paraboloid and hyperbolic paraboloid. The 3×3 SOM was trained using the feature set SC trained with 100 coarse training epochs.

To illustrate the differences and similarities between the three exemplar paraboloids we plotted the BMU histogram and views of the shapes coloured by vertex classification in Figures 5-2, 5-3 and 5-4 and the mean squared distance between each of the distributions

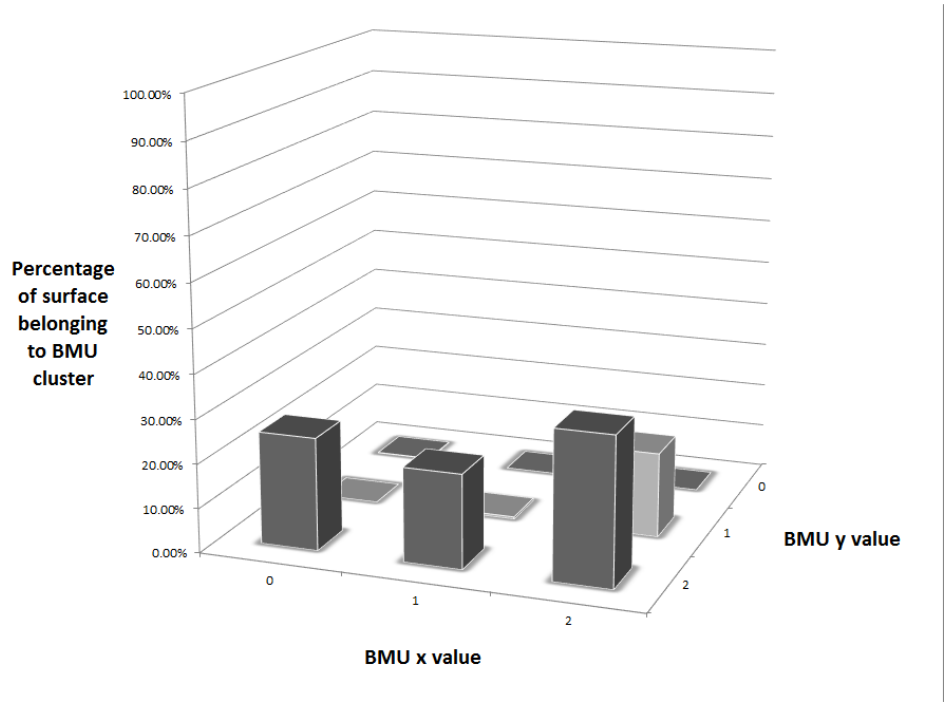


(a) BMU Distribution for Paraboloid

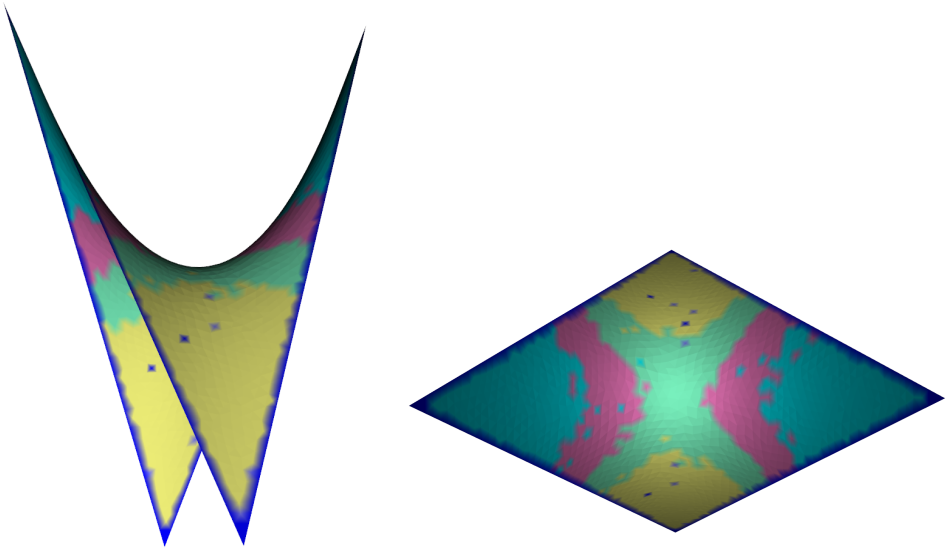


(b) Surface rendered by vertex cluster membership

Figure 5-2: Histogram and cluster membership visualisation for the Paraboloid

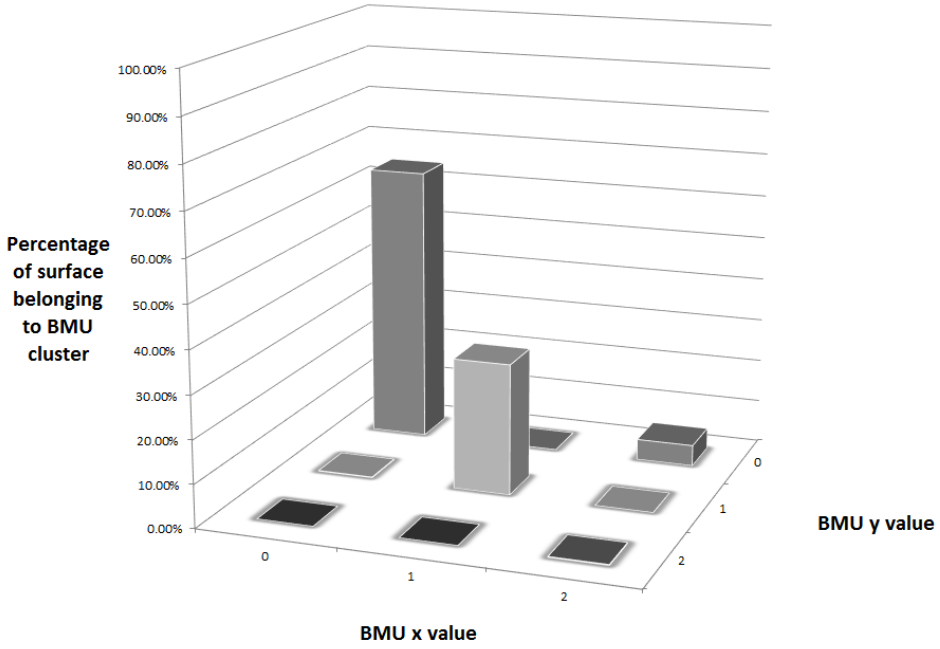


(a) BMU Distribution for Hyperbolic Paraboloid

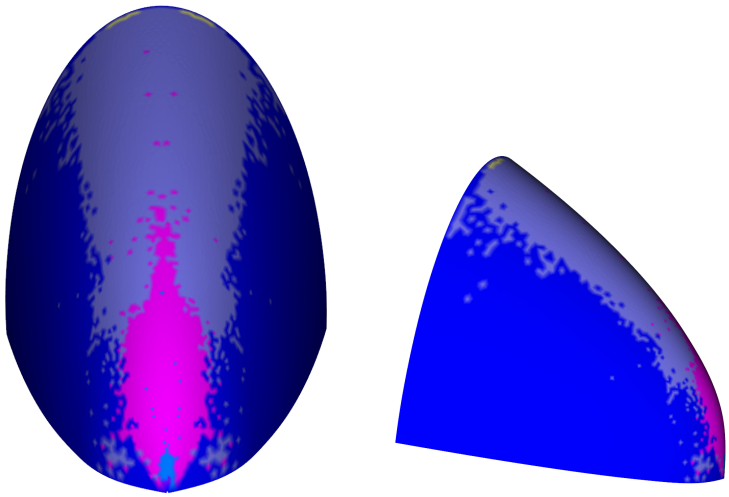


(b) Surface rendered by vertex cluster membership

Figure 5-3: Histogram and cluster membership visualisation for the Hyperbolic Paraboloid



(a) BMU Distribution for Elliptic Paraboloid



(b) Surface rendered by vertex cluster membership

Figure 5-4: Histogram and cluster membership visualisation for the Elliptic Paraboloid

S_1	S_2	Distance(S_1, S_2)
Hyperbolic Paraboloid	Paraboloid	0.12
Elliptic Paraboloid	Paraboloid	0.84
Elliptic Paraboloid	Hyperbolic Paraboloid	0.85

Table 5.1: Distances between BMU distributions for the exemplar shapes using mean squared difference

in Table 5.1. The mean squared difference results shows that the BMU distributions of the elliptic paraboloid (Figure 5-4(a)) and paraboloid (Figure 5-2(a)) share some commonalities and the hyperbolic paraboloid has a very different distribution (Figure 5-3(a)). These differences and similarities are apparent when visually inspecting the surfaces when the vertex clustering is visualised both the elliptic paraboloid and paraboloid share purple regions that represent medium curvature in one direction and a small curvature in the other (Figures 5-4(b) and 5-2(b)) and pink regions that represents similar curvature in both directions. The hyperbolic paraboloid represents a very different range of curvatures with a mix of concave, convex and near-flat regions. The results for the elliptic and regular paraboloids are similar for this feature set. This is not suprising as this 3×3 SOM represent only nine different curvature types. We expect these surfaces to become less similar as the number of neurons increases as they are constructed from different functions and represent different types of conic sections.

To describe regions or portions of an object in a consistent manner we will use a sphere as our selection widget. A sphere is used because it has a constant distance from the centre to its boundary. For these experiments the radius of the sphere is fixed and the centre is placed at a vertex of interest when searching for potential matches of equally sized regions. We investigate how experimental parameters such as feature sets, SOM sizes and training parameter variations influence the effectiveness of our approach.

Our system uses the object-relational mapping framework OpenJPA Foundation [2011c], Richardson [2008] to store object classes and instances in the Java based database Apache Derby [Foundation, 2011b]. This allows us to store every piece of data used in our exper-

Figure 5-5: Sample surfaces from shape database

iments such as surfaces, vertex based features, SOMs, sphere bound region definitions and region distributions in a system that enables simple persistence and retrieval of data. By reducing our indexing and retrieval problem to histogram matching we take advantage of the SQL engine's query optimization and indexing properties to quickly process large sets of data.

5.1.1 Face Data Set

For this set of experiments we will use range data representations of scanned faces taken from Moreno and A.Sanchez. [2004] and meshed using GMSH [Geuzaine and Remacle, 2003]. This is an interesting data set to use as faces are complex and the source data is noisy, making it difficult to extract curvature metrics. The surfaces are dominated by regions of continuous curvature that make finding consistent boundaries very difficult. Faces are also inherently symmetrical, each feature on the left side of the face using a matching feature on the right side of that same face, making this a natural choice for testing our ideas. A sample of the data set is illustrated in Figure 5-5, with an example of noise that can occur in the dataset illustrated in Figure 5-6, which may affect the region matching process.

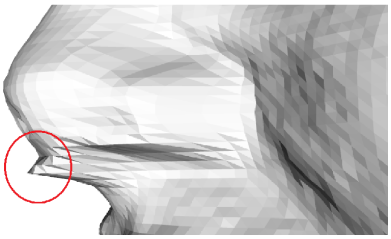


Figure 5-6: Noise in model from face database

5.1.2 Self-Organizing Maps

We are using the SOM to classify vertices and its response is encoded as a 2D histogram used to find similar regions. The eye-socket on the face in Figure 5-7 is our search region and we are expecting this search region to match at least one other region as the human face is largely symmetrical and the neural response should be similar to the opposing eye-socket region. This approach is similar to the histogram of oriented gradients technique [Dalal and Triggs, 2005] where global feature sets for regions without any encoded spatial relationships within the sets are used to recognise objects in a scene.

We will explore the indexing performance of different combinations of curvature metrics, SOM sizes and training epochs. Initially the dimensions of the SOM will be fixed at 20×20 while evaluating the effectiveness of different training data sets and feature combinations. The SOM size will be changed to 5×5, 10×10 and 30×30 using the most effective training data by our results for the 20×20 SOM.

5.1.3 Training Data

Before a SOM can be used as a classifier it requires training. The training data determines how the SOM will respond to the curvature values passed in as a feature vector. We will use three different sources of training data to see how well each data set works when searching for regions on a face taken from the GavabDB dataset. This dataset con-

tains very noisy range scans of human faces that have been remeshed into triangulated surfaces with very little additional processing making it difficult to identify matching regions.

The three training data sets are:

1. The complete range image database of 549 faces. The features are taken from all vertices in the database and fed through the SOM. This data set is used as it represents the type of data we wish to search at a later date. Some sample models from the face database are rendered in Figure 5-5.
2. The user selected search region shown in Figure 5-7. This training data uses the idea of self-similarity. The triangulated mesh is used as training data for the SOM.
3. A set of exemplar surfaces that represent different conic sections that can be fitted to many different curved regions on a free form surface. The three surfaces in Figure 3-1 on page 56 were created using the parametric equations in Table 3.1 on page 56. In Figure 5-10 the similar colouring of regions on the paraboloid and elliptic paraboloid illustrates how the SOM recognizes similarly curved regions.

5.1.4 Indexing a single face using distributions

In this Section we are using the search region in Figure 5-7 as the opposing eye-socket provides an ideal match to seek due to its similarity. The results of the search algorithm will be evaluated by manually inspecting the top 30 results for each feature, training data and SOM size variation.

The ranking in our tables of results refers to the distance of the descriptors generated using the SOM in comparison to all of the other distributions generated around each vertex of that face. The ranking of match values refers to the ranking of a match that covers the opposing eye socket when searching for a match on the same surface. For all

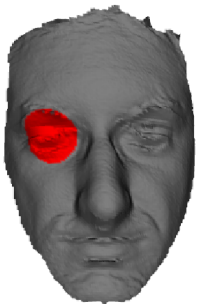


Figure 5-7: User selected search region marked in red

of our results this search region is found first. Results in this Section show what position in the top 30 matches the opposing eye-socket is found, an ”x” signifies that there was no match in the top 30. The search region coloured in red in Figure 5-7 is always the highest ranked match by definition. It is the rank of the opposing eye-socket that we are seeking. A low number of unique neurons being fired suggests that the SOM may not be able to sufficiently describe the variations of surface curvature, or that the surface itself is uninteresting in terms of its curvature.

5.1.5 Train on the User Selected Search Region

We trained the SOMs on the search region marked in red in Figure 5-7 with different sets of curvature based features. The results were poor (Table 5.2) with only three feature sets finding the opposing eye-socket within the top 30 matches returned by our region matching algorithm when looking for matches on this single surface. A lack of variation in the SOM BMUs being fired tells us that that only a small proportion of the SOM contains neurons that are representative of vertex features being classified.

The face in Figure 5-8(a) is coloured by vertex classification using the k_1 and k_2 curvature metrics as features and shows that the SOM is not able to detect the differences between regions that to the human eye have distinct changes in curvature. Qualitatively shown by the lack of colour variation using the BMU colour map (Figure 5-9) in the rendered

Table 5.2: Results of Search for training on Search region for a 20×20 SOM

Features	Ranking of match	Unique Neurons Fired	Percent Neurons Fired
$K\ H\ S\ C\ k_1\ k_2$	x	16	4
$K\ H\ S\ C$	x	36	9
$K\ H\ S$	x	15	3.75
$K\ H\ C$	x	6	1.5
$k_1\ k_2$	16	2	0.5
$K\ H$	27	8	2.0
$S\ C$	x	171	42.75
$H\ C$	23	44	11

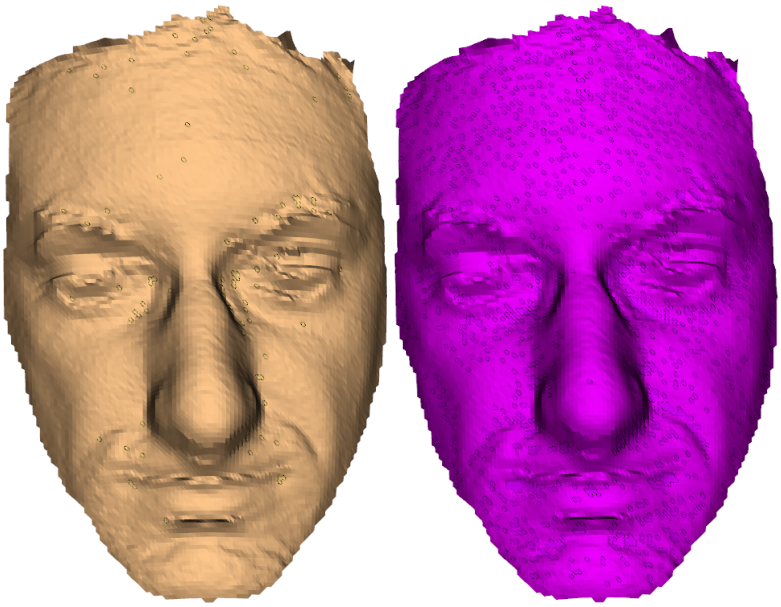
model, only 2 (0.5 percent) of the possible neurons fired during the classification process and 99 percent of the surface area of the model was represented by one neuron.

5.1.6 Train on all Faces in GavabDB

We trained SOMs using all of the meshed faces from the GavabDB [Moreno and A.Sanchez., 2004] data set. This data set consists of 549 depth maps of the faces of 61 different people of varying age groups with 45 male and 16 females taken at different angles with different expressions being shown. Triangulated meshes were then generated from these scans using GMSH [Geuzaine and Remacle, 2003].

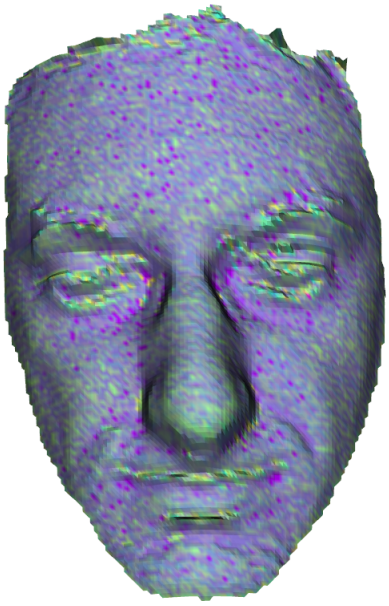
The results of this training set in Table 5.3 are similar to the poor results of the user selected search region data set and the search for the matching eye socket is successful for two of the seven features from this dataset.

Our best result for this training data in Figure 5-8(b), $k_1 k_2$, shows a lack of diversity of vertex types found by the classifier. This is an indication that only a small number of neurons in the SOM are needed to represent the vertex types of interest, with 5 out of a possible 400 different vertex types used to represent the face.



(a) Trained on User Selected Search region

(b) Trained on Whole GavabDB



(c) Trained on Exemplars

Figure 5-8: Images of best result for each of the three SOM training techniques using the neuron to colour mapping of Figure 5-9

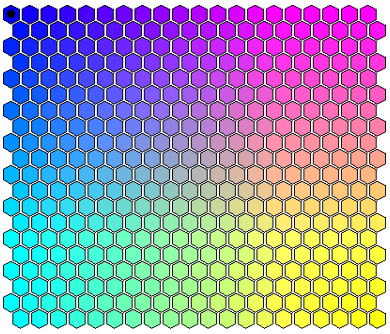


Figure 5-9: Colour palette for 20×20 SOM

The average number of unique feature vectors found by each of the feature sets has increased in comparison to the previous data set. This data set fired on average 45 unique neurons compared to the 36 for the SOMs trained on the search criteria. The increased size of the training data set provided a greater variety of values for each of the feature sets being fed into the SOM.

We had expected to find the search region in more of the feature combinations used in the experiment as the training set was much larger than the search criteria used in the previous experiment. This due to the excessive noise in the training data set which reduced the effectiveness of the SOM.

Table 5.3: Results of Search for training on all surfaces in the DB for 20×20 SOM

Features	Ranking of match	Unique Neurons Fired	Percent Neurons Fired
$K\ H\ S\ C\ k_1\ k_2$	x	21	5.25
$K\ H\ S\ C$	x	44	11
$K\ H\ S$	25	17	4.25
$K\ H\ C$	x	13	3.5
$k_1\ k_2$	2	3	0.75
$K\ H$	x	8	2
$S\ C$	x	211	52.75
$H\ C$	x	372	93

5.1.7 Train on Exemplars

The exemplar training data set had far more consistent results compared to the other two techniques. The surfaces were meshed using their mathematical definitions (Table 3.1 on page 56) to describe them as NURBS and then converted into a mesh representation. This training data is by definition low in noise and the resultant SOM performed far better than the SOMs trained using the search criteria and complete face data set with 4 of the 7 feature combinations having matches in the top 30 results and 3 for the KHS data set.

The mean number of unique feature vectors fired by the SOM during the training process (Table 5.4) is similar to the number fired when using the whole database and more than when the SOM was trained using the search criteria. This data set also produced a SOM that was able to consistently find the search region using a number of different feature sets with a lower variance in the number of neurons fired across the feature sets compared to the whole database training set. The best feature set was the combination of K, H and S. There were many instances of results that covered half of the opposing eye-socket and the same for matches to our search region.

Table 5.4: Comparison of neurons fired for each training data set

Training data set	Mean number of neurons fired	Standard deviation
User selected region	37	56
Whole DB	86	134
Exemplars	72	54

The colours used to classify the vertices on the exemplar training data set (Figure 5-10) can be matched to regions on the classified face in Figure 5-8(c). The dark purple region on the bridge of the nose is similar to the regions on the paraboloid and elliptical paraboloid where curvature is dominant or high in one direction and low in the other.

Table 5.5: Results of Search for training on exemplar surfaces for 20×20 SOM

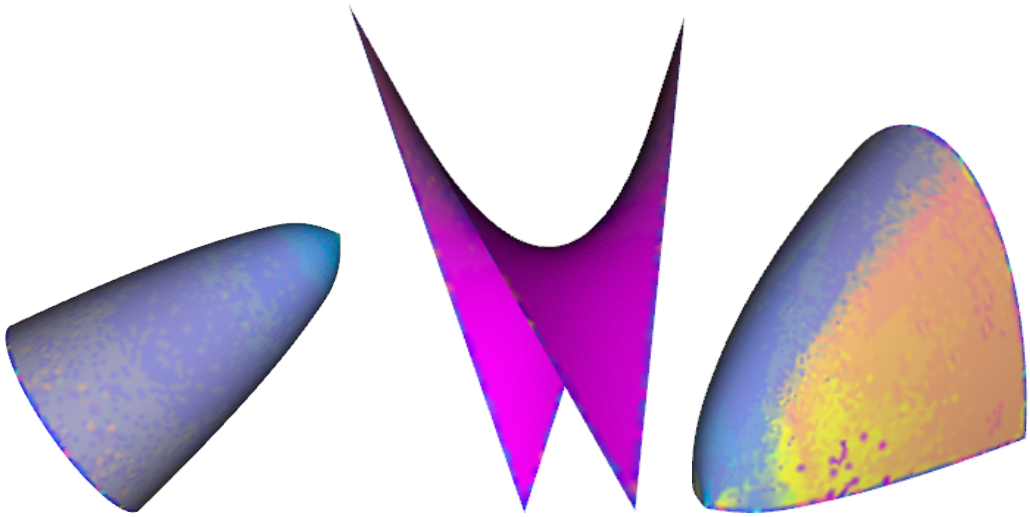
Features	Ranking of match	Unique Neurons Fired	Percent Neurons Fired
$K\ H\ S\ C\ k_1\ k_2$	x	104	26
$K\ H\ S\ C$	3	103	25.75
$K\ H\ S$	3, 4, 13	131	32.75
$K\ H\ C$	x	17	4.25
$k_1\ k_2$	22	16	4
$K\ H$	x	141	35.25
$S\ C$	14	42	10.5
$H\ C$	9	134	33.5

5.1.8 Variation of Self-Organizing Map Size

For the simple template matching process of seeking the opposing eye the exemplar training data was the better of the three sets so we will use it to investigate the effect of varying the size of the SOM we will use that training data set.

As the SOM size dropped the results in Tables 5.6 and 5.7 shows that the number of unique vectors decreased across all of the features. These smaller SOMs performed better with less features being used and the larger SOMs were better when all of the features available were used with the number of unique feature vectors also increased markedly.

The effect of increasing and decreasing the SOM dimensions can also be seen in Figure 5-11. The output from these SOMs illustrates the problem of noise in the data being indexed, regions that should be relatively smooth such as the forehead are not. From the results of varying the SOM size we can see that if the SOM is too large there are too many possible unique clusters, introducing more noise into the results of the classification making region matching more difficult.



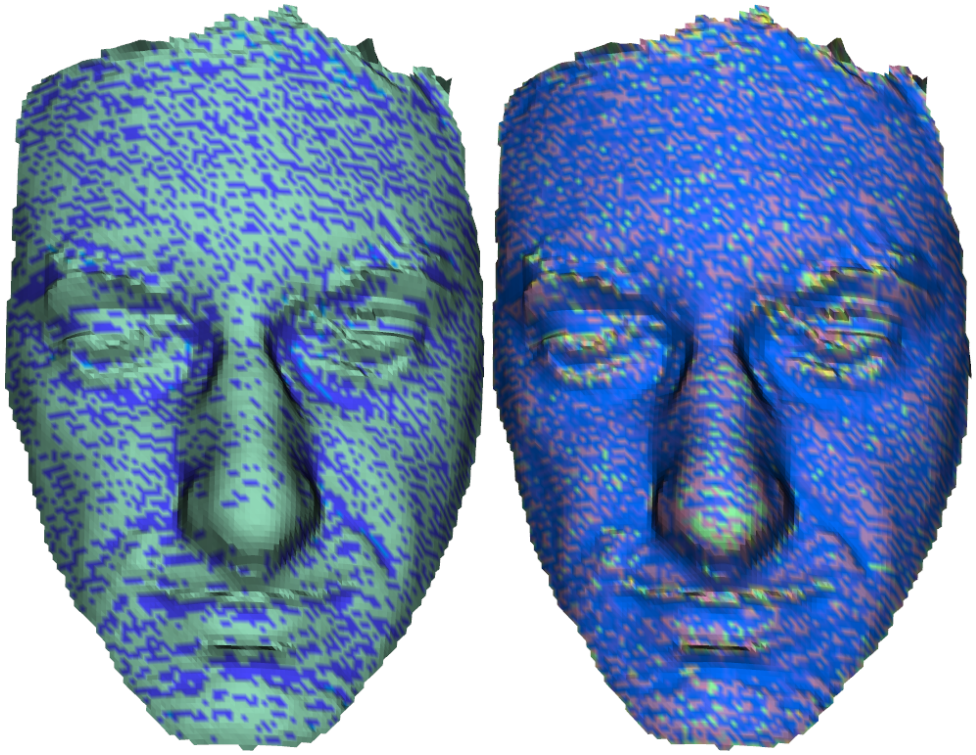
(a) Paraboloid (b) Hyperbolic Paraboloid (c) Elliptic Paraboloid

Figure 5-10: Exemplar surfaces coloured by a 20×20 SOM using K, H, S

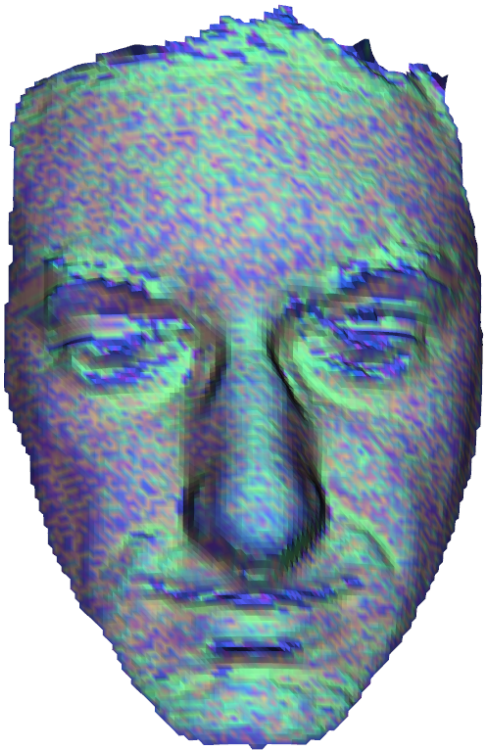
5.1.9 Discussion

Out of the three training data sets; (1) training on the surface of interest, (2) training on the whole database of surfaces, and (3) training on a small set of exemplar parametrised surfaces, the exemplar surface data set illustrated in Figure 3-1 using K, H and S consistently gave better results. This combination of training data and curvature features found the opposing eye-socket many times within the top 30 results. The effectiveness of exemplar surface training data set is due to low levels of noise in the training data compared to the other two sets. It may also be that it produces SOMs that are better suited to the histogram matching method used to find similar regions as it does represent a variety of paraboloids that can express the different components of the face, e.g. the bridge of the nose represented as a paraboloid.

When the size of the SOM was reduced the best results were produced when less features were used. As the SOM increased in size the number of features used for the best set of results increased. This suggests that the smaller SOMs were able to sufficiently express the different feature combinations when the feature count was low but as more features were used in the training process a larger SOM was required to effectively represent



(a) 5×5 $k_1 k_2$ (b) 10×10 HKS



(c) 30×30 K H S C k_1 k_2

Figure 5-11: Images of best results when varying the SOM size and trained on the exemplars

Table 5.6: Results of Search for training on all exemplar surfaces for 5×5 SOM

Features	Ranking of match	Unique Neurons Fired	Percent Neurons Fired
$K\ H\ S\ C\ k_1\ k_2$	x	3	12
$K\ H\ S\ C$	8	4	16
$K\ H\ S$	10, 11	4	16
$K\ H\ C$	25	4	16
$k_1\ k_2$	2, 12	4	16
$K\ H$	8, 10, 20	5	20
$S\ C$	24	5	20
$H\ C$	15	9	36

the greater number of feature combinations. Conversely when a smaller SOM was used with a large feature set it didn't have enough neurons to sufficiently express the different feature combinations in the training data.

When the SOMs were trained using the search region the number of unique vertex types was very low. As only a few different features are used to train the SOM due to the small number of vertex types within it, the SOM's ability to cluster different vertex types was compromised due to its limited training data set.

When the k_1 and k_2 features are used together without any other features the number of unique feature vectors is very low regardless of the training data set used. In most of the experiments in this Section these feature pairs in isolation were able to be used in the training and clustering process to generate a SOM that was able to find the opposing eye. The surfaces processes in this section were too noisy to produce results of significance and our technique appears to be too sensitive to noise to be applied successfully to range data. This is supported by our observations in Section 2.2.2 where we found the discrete curvature measurement technique to be very sensitive to noise in the surfaces. The SOM has no way of discerning noisy data from valid input so it attempts to organize its neurons to fit all incoming data regardless of how we perceive the surface data when visualised.

Table 5.7: Results of Search for training on all exemplar surfaces for 10×10 SOM

Features	Ranking of match	Unique Neurons Fired	Percent Neurons Fired
$K\ H\ S\ C\ k_1\ k_2$	x	29	29
$K\ H\ S\ C$	8, 20, 22	29	29
$K\ H\ S$	2, 29	41	41
$K\ H\ C$	5	6	6
$k_1\ k_2$	x	7	7
$K\ H$	x	39	39
$S\ C$	26	15	15
$H\ C$	x	27	27

5.2 CAD Models

In the previous Section we processed a noisy dataset of 3D models generated from multiple range scans. In this Section we attempt to match regions on clean triangulated meshes extracted from IGES models of various sizes (Figure 5-12) using a meshing program [CIMME, 2007]. We trained the SOM used to classify the vertices of these models shown with the exemplar paraboloids and the training parameters that gave us the best indexing from Section 5.1.

The KHS feature set gave consistent results (Section 5.1.9, Figure 5-11(b)) so we used these features and a 10×10 SOM to classify the vertices used to build our index based on BMU distributions.

We searched for the region marked in red in Figure 5-13 which is similar to the Sections circled in yellow using 100 coarse training epochs and 0 fine training epochs. We expect to find some of these regions in the top 30 results using this combination of features as they performed consistently well for the face dataset (Section 5.1.9). We reduced the number of duplicate results by removing results that shared 75 percent of the same vertices of any

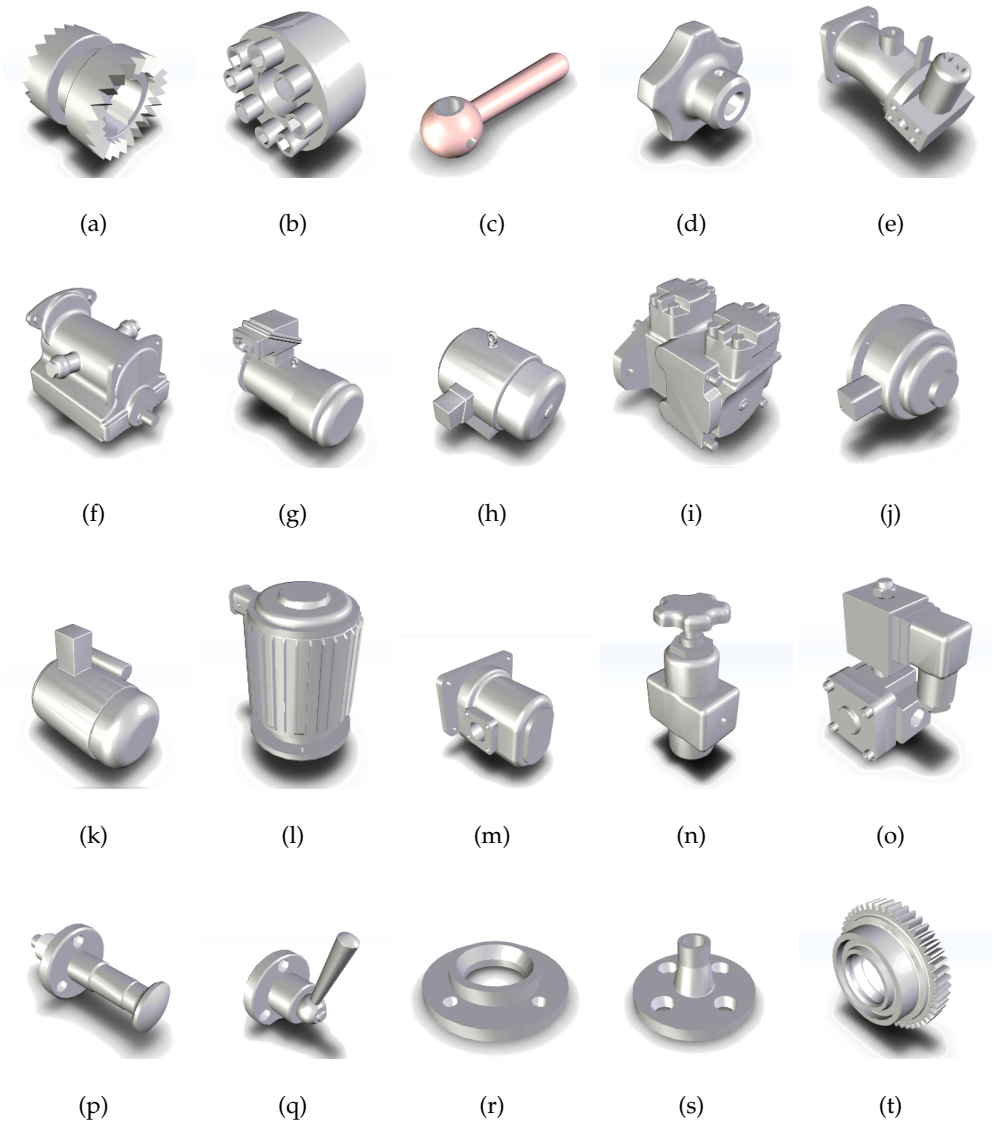


Figure 5-12: 3D CAD Models rendered using their IGES representation [New Dimension Systems, 2011]

Table 5.8: Results of Search for training on all exemplar surfaces for 30×30 SOM

Features	Ranking of match	Unique Neurons Fired	Percent Neurons Fired
$K\ H\ S\ C\ k_1\ k_2$	10	196	21.77
$K\ H\ S\ C$	x	177	19.7
$K\ H\ S$	x	20	2.22
$K\ H\ C$	x	208	23.11
$k_1\ k_2$	x	26	2.88
$K\ H$	x	238	26.44
$S\ C$	x	55	6.11
$H\ C$	2	247	61

existing results in the set.

We found that for our 10×10 SOM using the features KHS we didn't find any of the expected regions marked in yellow in Figure 5-13. Instead we found regions that included slightly curved sections and sharp edges present in our search criteria, but not arranged in the same way. This is not unexpected as our bag of words approach does not encode spatial relationships. To understand why this set of SOM parameters gave these search results we examined the 3D model after colouring it based on how the vertices were classified by the SOM.

The results in Figure 5-15 show that the SOM was unable to distinguish between flat regions, sharp edges and corners. Without being able to distinguish between these basic features there is little chance that this SOM would perform well as the basis of our indexing and retrieval system. To confirm that this was the case for other models in our database we classified the vertex features of each of the other models in the CAD database and then segmented them by connecting vertices that shared the same neuron and found that planar regions that shared a common sharp edge were a part of the same patch. The classified vertices and segmented regions for some of these are rendered in Figure 5-16, the light coloured regions that have been circled represent small connecting planar patch

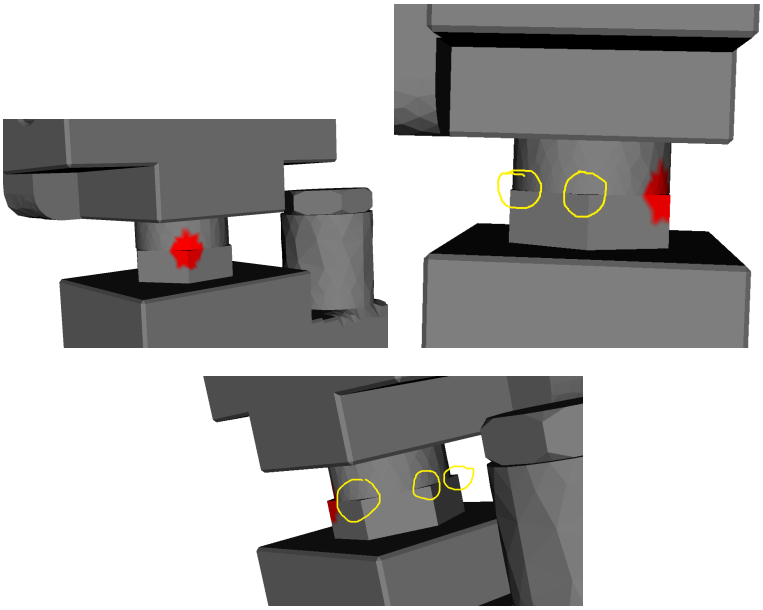


Figure 5-13: Selected region from model in Figure 5-12(o) used as search criteria marked in red and visually similar regions circled in yellow

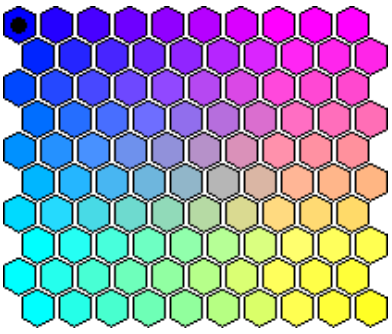
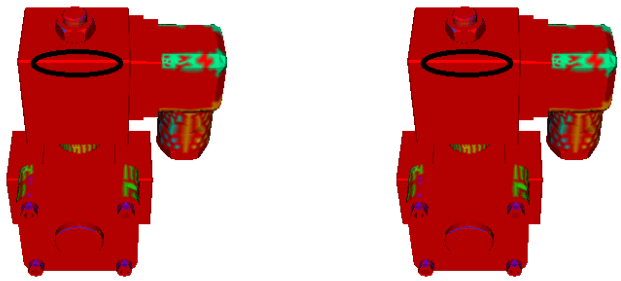


Figure 5-14: Colour palette for 10×10 SOM.



(a) Classified vertices (b) Connected components

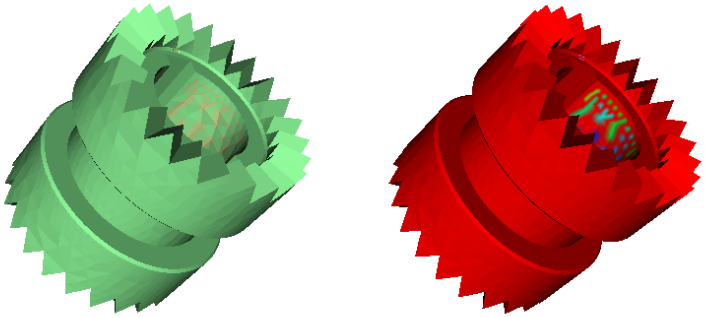
Figure 5-15: CAD model classified using KHS as features on a 10×10 SOM trained for 100 epochs with examples of rendering artefacts marked.

and are a different colour due to an artefact of the model rendering process.

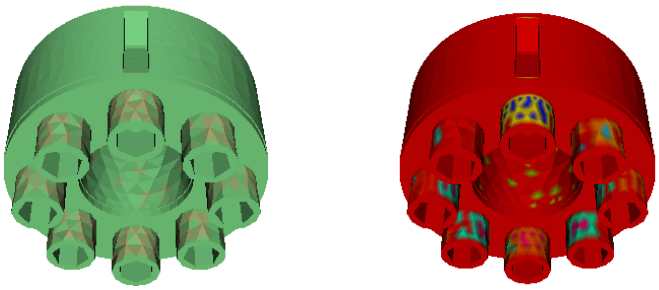
These results suggest that SOM training data, features or training iterations used for the noisy face dataset (Section 5.1.7) aren't appropriate for the CAD dataset we are now indexing. We increased the SOM size to 20×20, keeping the experimental parameters the same and encountered the same problem. Normally increasing the SOM size would increase the number of distinct regions to be found on the model. For the CAD training set this didn't hold (Figure 5-17) so we tried alternate feature combinations.

The model used in Figure 5-15 was used to evaluate the effectiveness of each of the feature sets as the results of classifying the vertices was representative of how well the SOM identified salient features in other models in the database (Figure 5-16). The feature combinations used are:

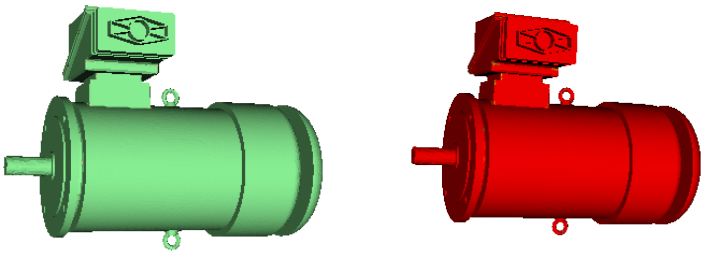
- $K\ H\ S\ C\ k_1\ k_2$
- $K\ H\ S$
- $k_1\ k_2$
- $K\ H\ S\ C$



(a) Classified vertices (b) Connected components

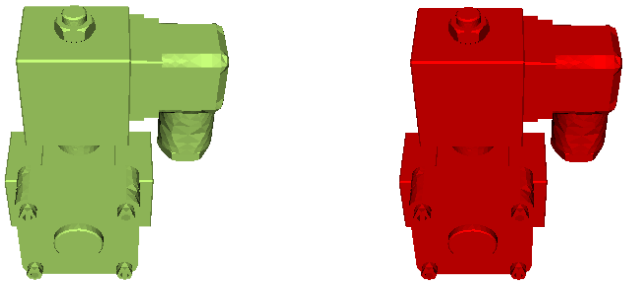


(c) Classified vertices (d) Connected components



(e) Classified vertices (f) Connected components

Figure 5-16: CAD models classified using KHS as features on a 10×10 SOM trained for 100 epochs, classified vertices coloured with the mapping shown in Figure 5-14



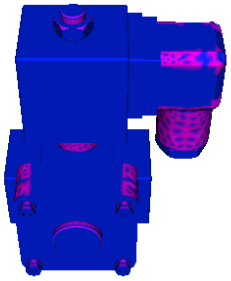
(a) Classified vertices (b) Connected components

Figure 5-17: CAD model classified using KHS as features on a 20×20 SOM trained for 100 epochs coloured using the palette in Figure 5-14

- $K\ H\ C$
- $K\ H$
- $S\ C$
- $H\ C$

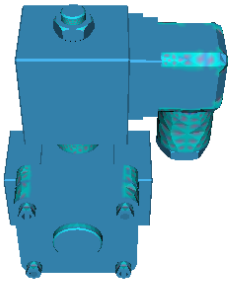
For each of the feature combinations the SOM was unable to detect critical features such as sharp corners and edges, a selection of these results is illustrated in Figure 5-18. Only the KHS feature set (Figures 5-18(d) and 5-18(c)) was able to identify two planar regions bordered by a curved edge.

When segmenting a CAD model (Section 4.4 on page 166) the most effective dataset for classifying vertices was the model being segmented, using 10 coarse training epochs and HC as the features. These parameters didn't work well for the noisy face dataset, but our 3D models have been created from CAD models represented as IGES (NURBS) so the level of noise in the models is far less compared to the faces. We trained a 5×5 SOM with these training parameters to see if the concept of self similarity discussed in Section 3.2 on page 78 could apply to our CAD model indexing.

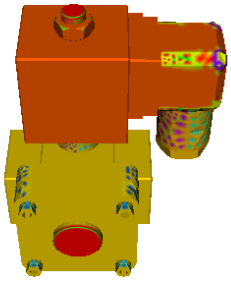


(a) $KHSC$ Classified vertices

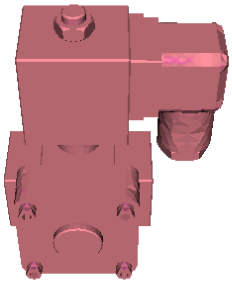
(b) $KHSC$ Connected components



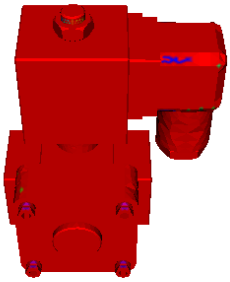
(c) KHS Classified vertices



(d) KHS Connected components



(e) KC Classified vertices



(f) KC Connected components

Figure 5-18: CAD model classified using various feature sets and a 10×10 SOM trained for 100 epochs, classified vertices coloured with the mapping shown in Figure 5-14 on page 195

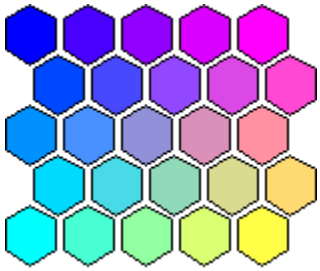
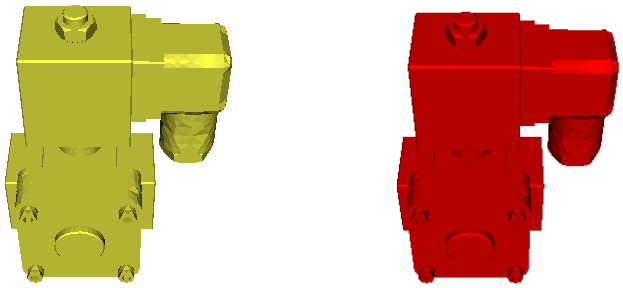


Figure 5-19: BMU to Colour Map for 5×5 SOM with the origin marked in the top left corner



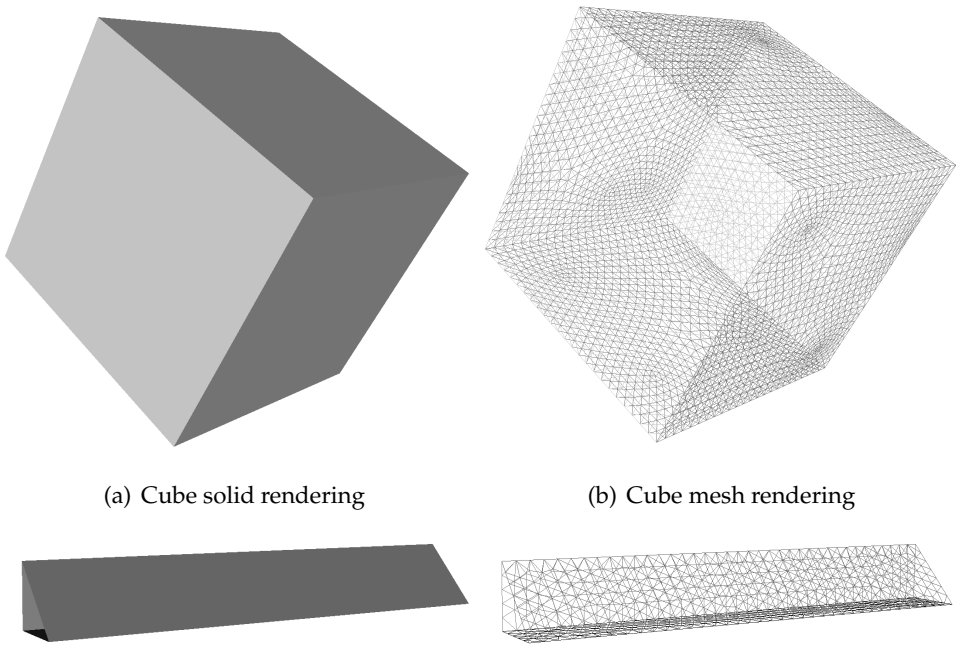
(a) Classified vertices (b) Connected components

Figure 5-20: CAD model classified using HC as features on a 5×5 SOM trained for 10 epochs. Classified vertices coloured using the mapping in Figure 5-19

The result of colouring the vertices of the model based on what cluster each vertex belonged to when the SOM was trained on all of the surfaces in the CAD model database (Figure 5-20) showed little variation. Flat surfaces were the same colour as edges and areas where there was curvature in one and both directions. The CAD models shown in Figure 5-12 are also dominated by flat regions and region with curvature in only one direction, making for few interesting regions with complex curvature changes. These preliminary results suggest the SOM training dataset needs features that describe flat surfaces and edges with sharp angles which aren't a part of the paraboloid training set.

To improve the SOM's ability to detect sharp edges we included a wedge (Figure 5-21(c)) and cube (Figure 5-21(a)) in the training data set to make the SOM sensitive to sharp angles in one direction, re-ran the index creation process and visualised the result of classifying the vertices using our new SOM. We increased our epsilon value from $1.0e^{-10}$ to $1.0e{-}5$ when calculating our curvature measurements which reduced the noise level on planar regions. See Section 2.2 on page 23 for the curvature calculation algorithms. It is important to note that for continuous surfaces curvature at a point is undefined but since we are using discrete techniques that uses the sum of angles around the point we can define curvature at the edges and corners of our exemplar surfaces.

We classified the surfaces of the additional exemplar shapes with SOMs trained with these additional shapes and only the original exemplars to visualise the impact of the additional training data. The results for the cube in Figure 5-23 shows little impact in terms of edge detection but the use of an alternative colour mapping highlighs clear examples of overfitting on the flat regions when the additional training data was used to train the SOM. The overfitting appears to have little impact on the results shown in Figure 5-24 where a CAD part was classified using the new exemplar set. It resulted in better patch identification compared to the results using the same training parameters and the original exemplar set in Figures 5-18(a) and 5-18(b) where none of the beveled edges were detected. Our new set of results shows better, but not perfect, edge detection with most of the 45 degree edges identified by our clusterer. This improved edge detection was a result of adding the wedge exemplar. The impact of this additional training data is shown



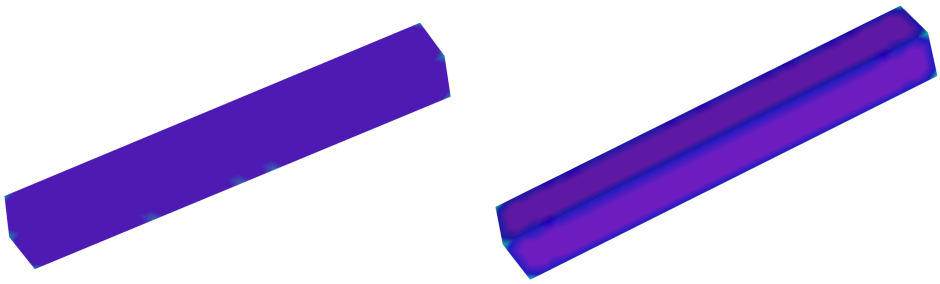
(a) Cube solid rendering

(b) Cube mesh rendering

(c) Wedge solid rendering

(d) Wedge mesh rendering

Figure 5-21: Additional exemplar surfaces to assist in detecting sharp angles in one direction



(a) Original exemplar set (b) Exemplars with cube and wedge

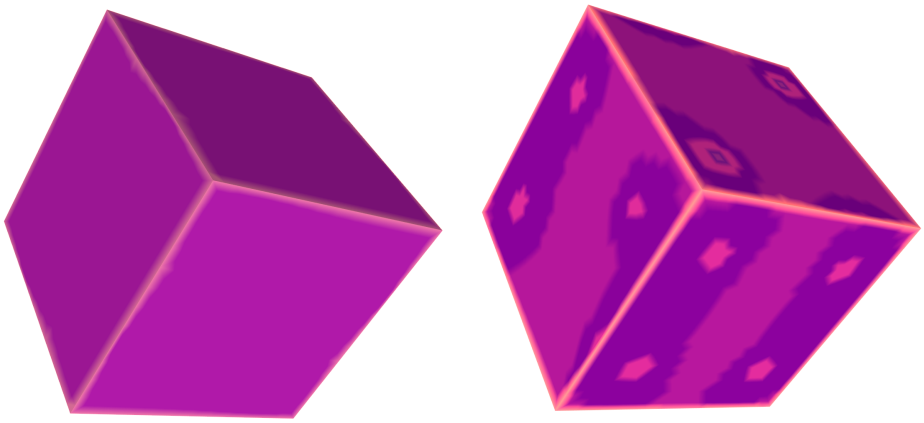
Figure 5-22: Comparison of vertex classification of wedge using exemplar sets with and without the cube and wedge.

Table 5.9: Feature combinations used to evaluate new exemplar training dataset

$KHSC$	KH	SC
k_1k_2	$KHSCk_1k_2$	KHk_1k_2
SCk_1k_2	KS	HS
k_1S	k_2S	Kk_1
Sk_1	Sk_2	KC
Kk_2	KHS	KHC
KHk_1	KHk_2	HC

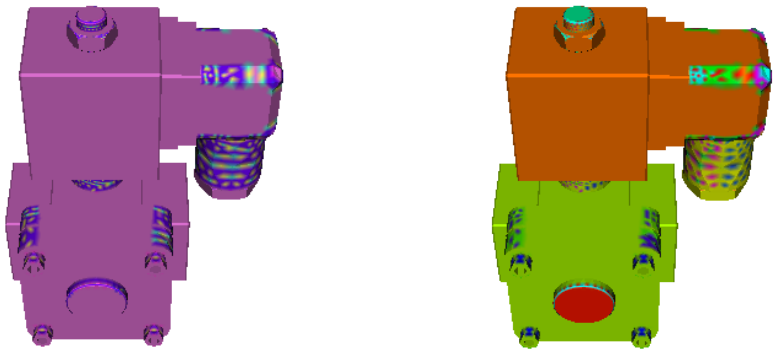
in Figure 5-22 where the original exemplar set failed to detect any of the edges of the wedge.

After making the changes to our training data set we trained SOMs using the feature combinations in Table 5.9 and found the feature combination $KHSC$ was able to detect most of the sharp edges on the model (Figure 5-25). These features are less sensitive to the absolute principal curvature metrics k_1 and k_2 that they are derived from and both KH and SC are able to describe a wide variety of surfaces [Besl, 1988, Koenderink, 1990]. We then varied the coarse and fine training epochs using combinations of 100, 10 and 0 for the fine and coarse training stages and used 5×5, 10×10 and 20×20 SOMs. The feature set $KHSC$ with a 10×10 SOM and 100 coarse training epochs was the best combination of parameters for detecting edges, curvature in two directions and planes as shown in



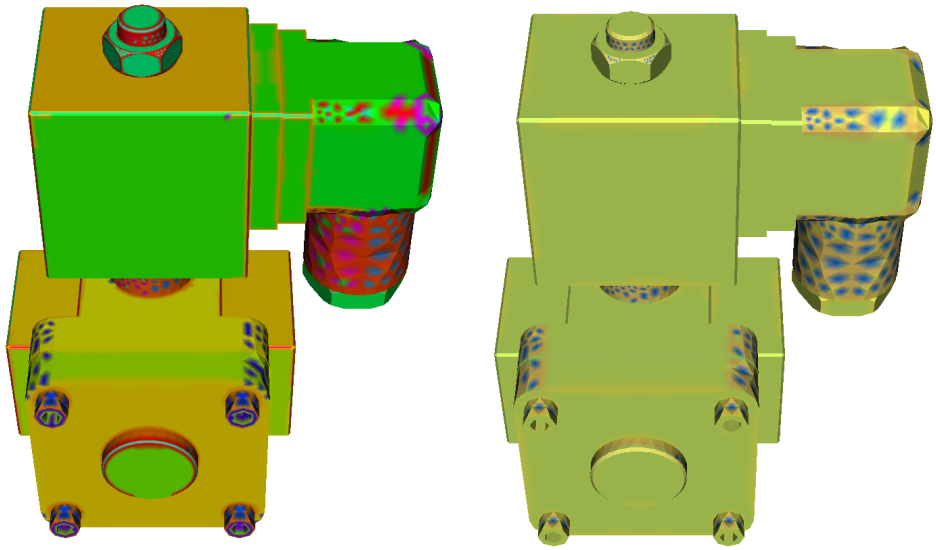
(a) Original exemplar set (b) Exemplars with cube and wedge

Figure 5-23: Comparison of vertex classification of a cube with an alternate colouring scheme using exemplar sets with and without the cube and wedge.



(a) Classified vertices (b) Connected components

Figure 5-24: CAD model classified using KHS as features on a 10×10 SOM trained for 100 epochs with new exemplar shape set and coloured with the mapping shown in Figure 5-14



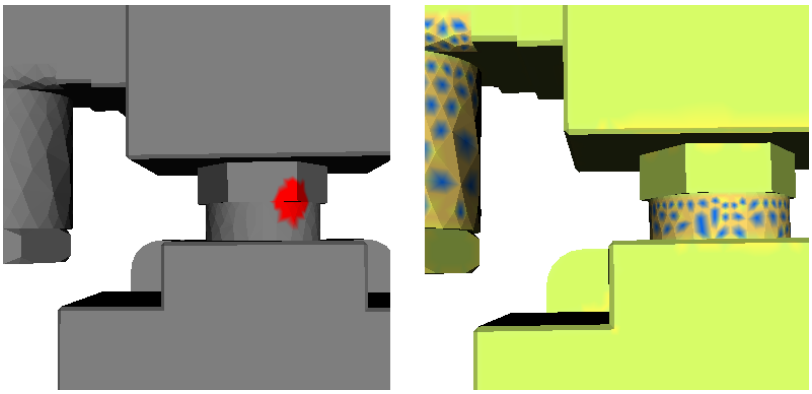
(a) Patches of connected components (b) Vertices coloured by cluster

Figure 5-25: Model using features $KHSC$, 10×10 SOM and 100 coarse training epochs and coloured with the mapping shown in Figure 5-14

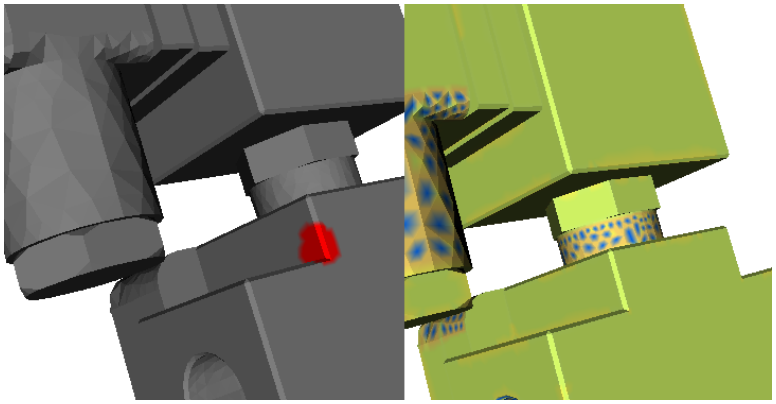
Figure 5-25(a) where the planar sections and edges that make up the cube-like shape on the top of the model are marked in different colours.

After matching regions based on BMU distributions we found that using the new set of exemplars as a training data set did not improve our search results for our test selection. Upon inspection the highest ranked match did contain the correct proportion of planar regions to sharp edges when compared to the search criteria but it wasn't until the 17th result that a match from a visually similar region was found. The results in Figure 5-26 highlights one of the weaknesses of our approach. If the salient features of a region aren't sufficiently represented within the BMU distribution (Figure 5-26(b) and search criteria histogram (Figure 5-27(a)) or there isn't sufficient variation in features within the region then we will not rank visually similar regions as highly as expected. The same can be said if the meshing algorithm doesn't handle curved regions consistently when converting from NURBS based models such as IGES files.

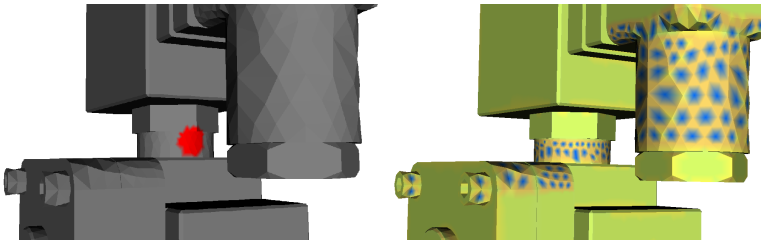
We then selected another region that had obvious matches and more variation in cur-



(a) Search region



(b) Top ranked match



(c) 17th ranked match

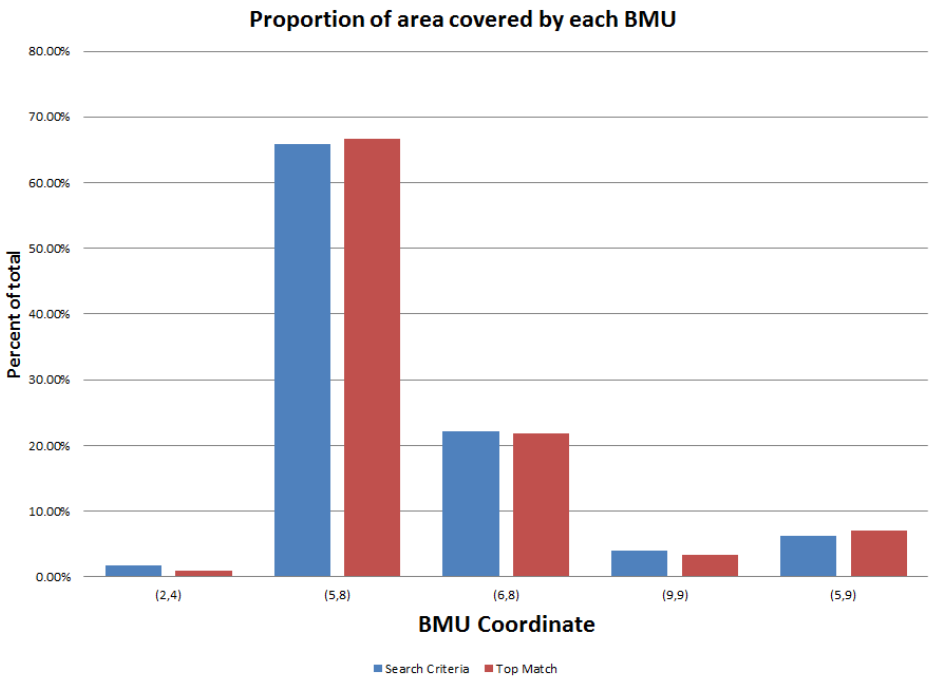
Figure 5-26: Search criteria and results (marked in red) finding match for region in Figure 5-26(a).

vature compared to our previous selection. This time we had excellent results with all matches within the top 5 bounding regions (Figure 5-28) that upon visual inspection (Figure 5-28(b)) appeared to be very similar and the BMU distribution in Figure 5-27(b) reflected the greater variance in curvature compared to Figure 5-27(a). From this experiment we can see that the greater amount of variation in curvature types that translates into more neurons being fired within the search region translates into a richer set of features that can be more accurately matched against other regions in the database.

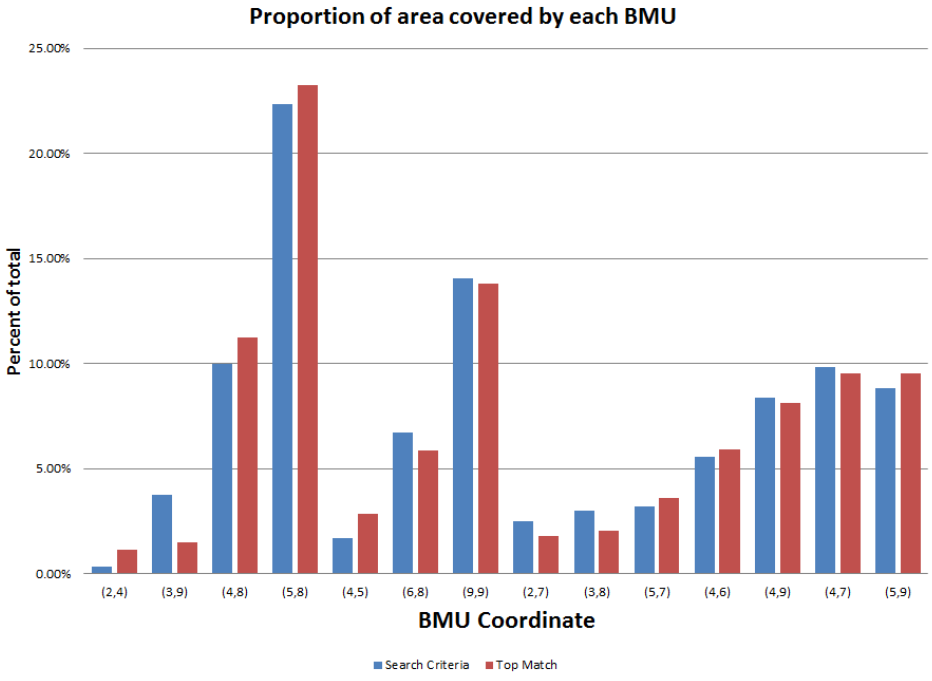
To test our technique across the whole CAD model database shown in Figure 5-12 we selected a region (Figure 5-29(a)) that has similar regions both on the that model and others in the database. The BMU distribution for our new search criteria (Figure 5-30) isn't as evenly spread as the descriptor for our previous experiment, but given how uniform the patches (Figure 5-29(c)) and vertex classification (Figure 5-29(b)) are for each of the regions on that shape that are similar in form, the set of results should include the other similar regions on that same model. Upon inspection the curved region of the search criteria (Figure 5-29(b)) is coloured in a similar pattern across all regions on the model that are convex and are curved in one direction. This suggests that the SOM may not be able to sufficiently discriminate between regions that are curved in a single direction. Plotting the distribution as a histogram (Figure 5-30) the selection is dominated by a single BMU (84 percent) and the rest of the distribution dominated by another neuron with a tiny proportion of the surface area represented by 8 other neurons.

Two searches were made, the first against the model that we selected the region from, using only regions that shared the same spherical radius as the search criteria object. The results of the first set (Figure 5-31) shows that our technique identified matching regions on the model as well as regions with a narrow edge section that joins two large sections with one convex and the other completely flat. The second search across the whole database (Figure 5-32) yielded regions that are very similar to the search criteria, including many sections that are almost identical to our search criteria.

We ran the same experiment using the Earth Movers Distance (EMD) function to calcu-

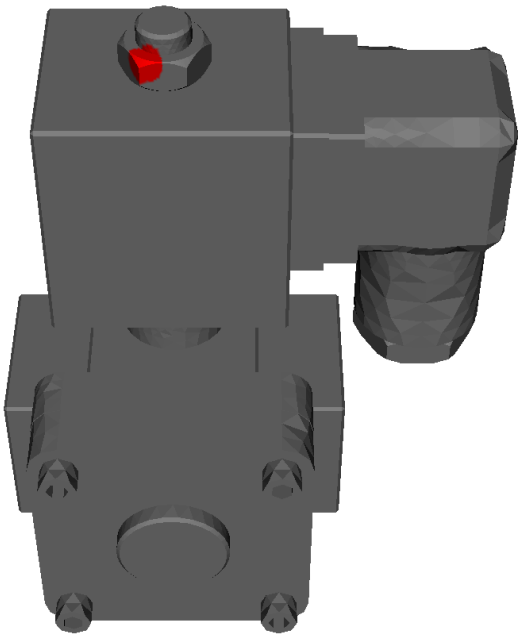


(a) Match in Figure 5-26(b) compared to search criteria

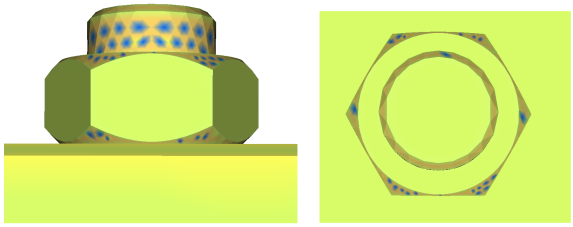


(b) Match in Figure 5-28 compared to search criteria

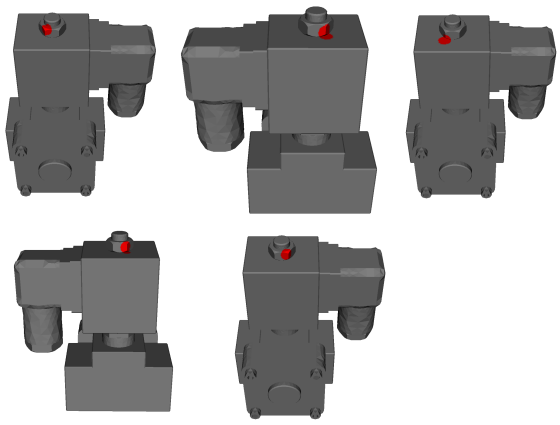
Figure 5-27: Comparison between search criteria and best matching BMU distributions



(a) Search region

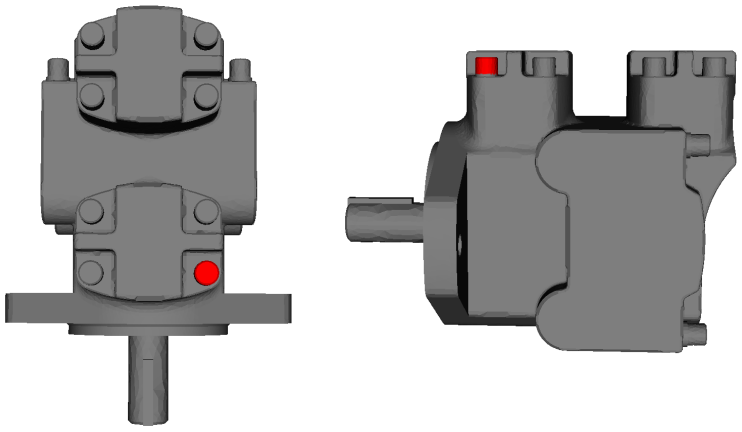


(b) Search region classified side and top view

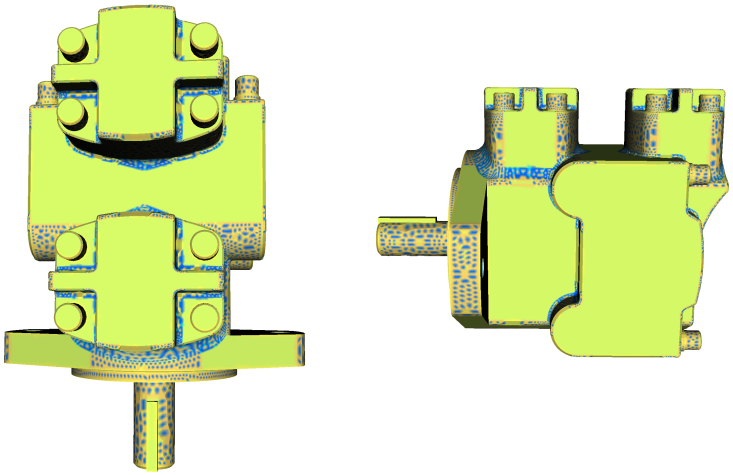


(c) Top 5 matches

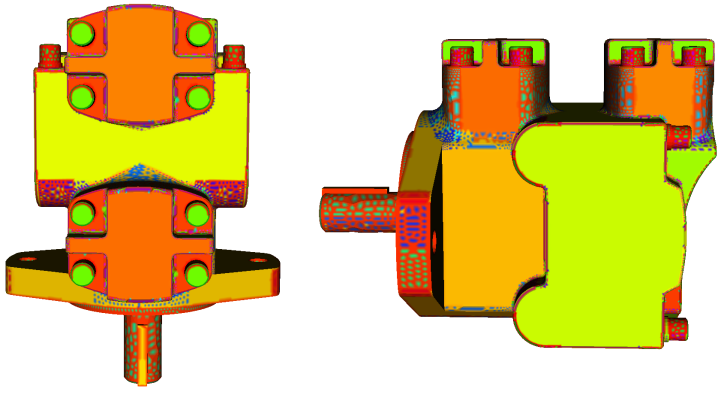
Figure 5-28: Search criteria and results for a region with a greater variation of curvature types and neurons



(a) Selection



(b) Object with classified vertices



(c) Patches of connected components

Figure 5-29: Search used for finding matching regions across whole DB

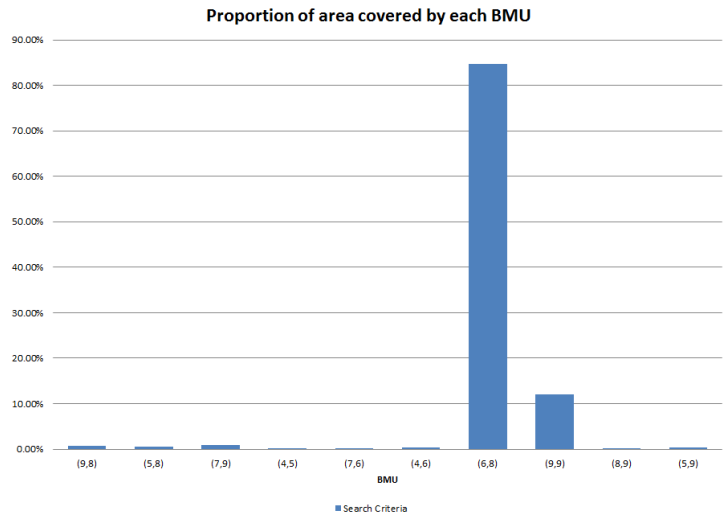


Figure 5-30: BMU Distribution for search criteria in Figure 5-29(a)

late the feature space distance between two distributions and found that there was very little difference between the results sets. This may be due to the relatively small number of neurons being fired, negating the benefit of the EMD's ability to take into account distances between each of the distributions when different neurons are fired. These results highlights the lack of spatial relationship encoding in our descriptor that reduces the effectiveness of our technique when dealing with relatively uniform surfaces.

The two previous searches used the same sphere radius as the search criteria (13 units), restricting out potential matches because scale becomes a factor due to the fixing of the radius and the lack of spatial feature encoding in our index. To find other matches we built an index of BMU distributions in our SQL backed object storage using sphere bound regions with radii of 5, 10, 15 and 20 units and used a SQL query to identify regions that contained a similar proportion of the region containing the dominant BMU of $(6, 8)$ (Figure 5-30). Using the regions identified with the original query we used a priority queue and the sum of squared differences between each of the regions to find our best matches.

We executed a SQL query to extract the regions of interest by looking for other region

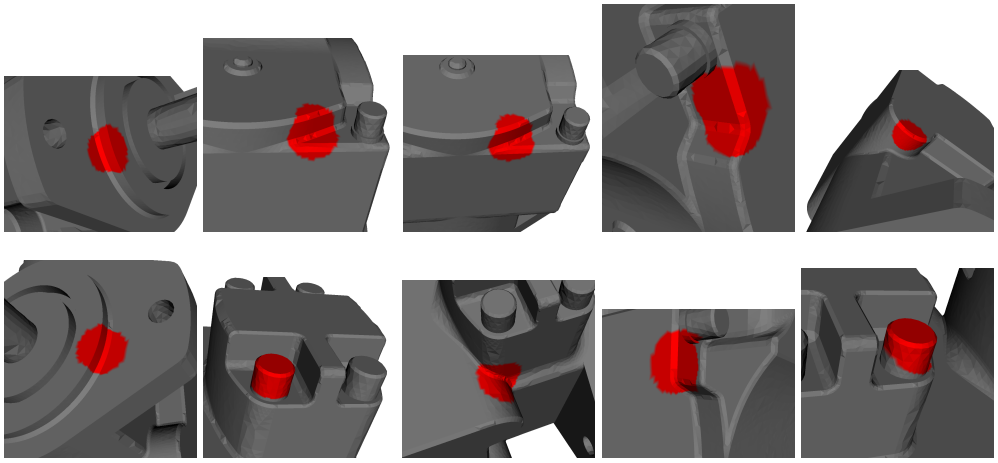


Figure 5-31: Top 10 results from model containing the search criteria in Figure 5-29(a) excluding the search criteria

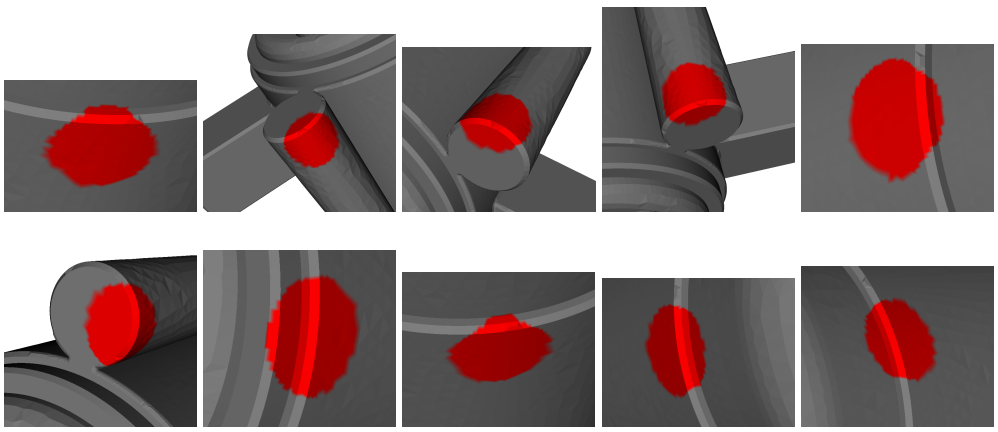
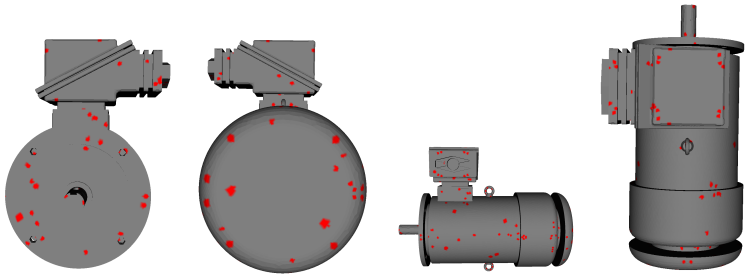


Figure 5-32: Top 10 results for matches with models in DB with fixed radius



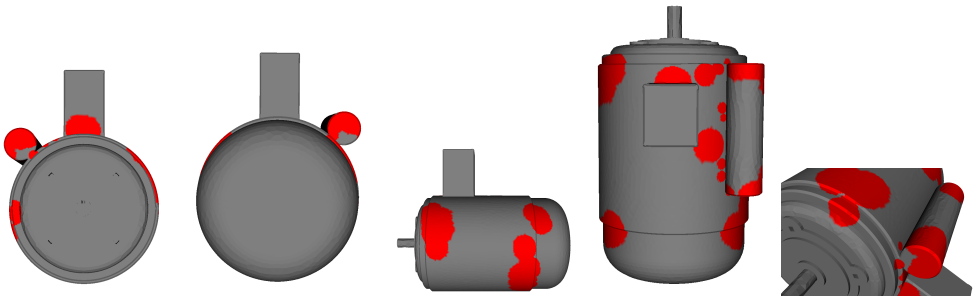
(a) Front (b) Rear (c) Side (d) Top

Figure 5-33: Surface matches in the top 500 results using SQL query for dominant BMU within 10 percent of search criteria

distributions where the dominant BMU was within 10 percent of the target proportion. There were 2 unique surfaces detected with results from each of the four sphere radii (5, 10, 15 and 20), returning 8 percent of the complete set of region distribution records. From this set of potential matches we then used the same technique as the previous Section and used the mean squared difference between the distributions and target distribution to rank the matches. We then marked the top 500 matches on the two objects that had regions that matched our search region, excluding the object that contained the search criteria.

The results (Figures 5-33 and 5-34) shows many matches of the correct shape but scale is still an issue when comparing regions. The sphere ranges of 5, 10, 15 and 20 are not sufficient to allow searching across all of the models in the DB. Figure 5-33 shows that the matching regions are much smaller than expected. We have good matching regions in Figure 5-34, they are visually similar to the search critera with regions curved in one direction along with the bevelled edge and flat section.

To take into account scale variation between these models we added more entries to the index using a number of larger sphere radii and searched for the same region (Figure 5-29(a)). After running the same search with the additional entries in the index we found that the top 10 results were the same, but we did see matches for another surfaces starting at the 12th ranked result (Figure 5-35) that showed similar attributes: two large flat



(a) Front (b) Rear (c) Side (d) Top (e) Detail

Figure 5-34: Surface matches in the top 500 results using SQL query for dominant BMU within 10 percent of search criteria

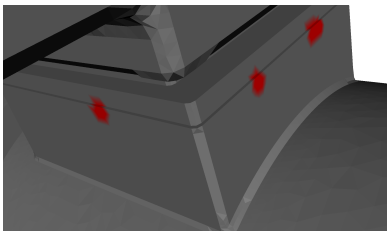


Figure 5-35: Surface matches in top 500 results using SQL query for dominant BMU within 10 percent of search criteria with more region radii in index

sections joined by a small flat section with two sharp angles.

5.3 Bimba Con Nastrino

We investigated the classification of vertices using the model titled Bimba con Nastrino in Section 3.2 on page 78 and now we will look for regions of self similarity by building an index of sphere radii that is the same as our search criteria in Fig 5-36. This region was selected because qualitatively it can be seen to be repeated in the model. Given that the best results for classifying a SOM, when the classified surface is to be compared to itself, we trained the SOM using the model as the training data and the same training parameters: i.e. 16×16 SOM using SC and 100 coarse training epochs, found to work

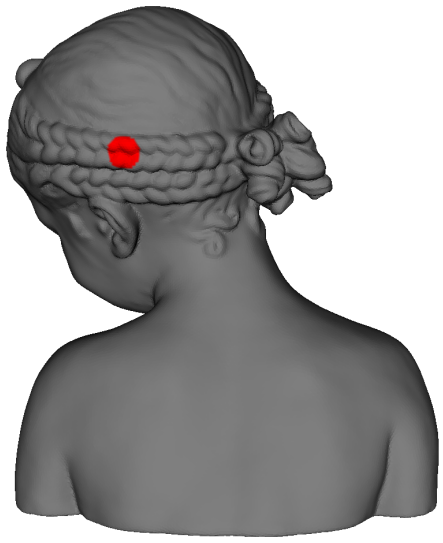


Figure 5-36: Search criteria for search within the model

well in Sec .3.2.3 on page 96.

The initial top 10 matches using the LMS and EMD (Figure 5-38) show regions in the results that are similar to the search criteria when visually inspected. When visualising the top 50 matches using the two distance calculations (Figure 5-39) we observe the benefit of using the more flexible EMD with more hits surrounding the search criteria (Figure 5-36) and other regions that are visually similar. The EMD is able to accommodate matching between regions because it takes into account the distance between clusters when comparing distributions compared to the mean squared difference which has no way to accommodate this. This is an important property of the EMD algorithm when applied to our indexing and retrieval task because of the large number of clusters represented in the index that are in some cases very similar to their neighbouring cluster. The set of results for EMD contained many matches clustered in small regions with overlapping results so we visualised the top 200 results in Figure 5-40 to illustrate how well the braid is identified by our search.

To increase the number of potential matching regions we added entries to the index with radii 10, 20 and 30 percent larger than the search criteria radius and re-ran the query. The

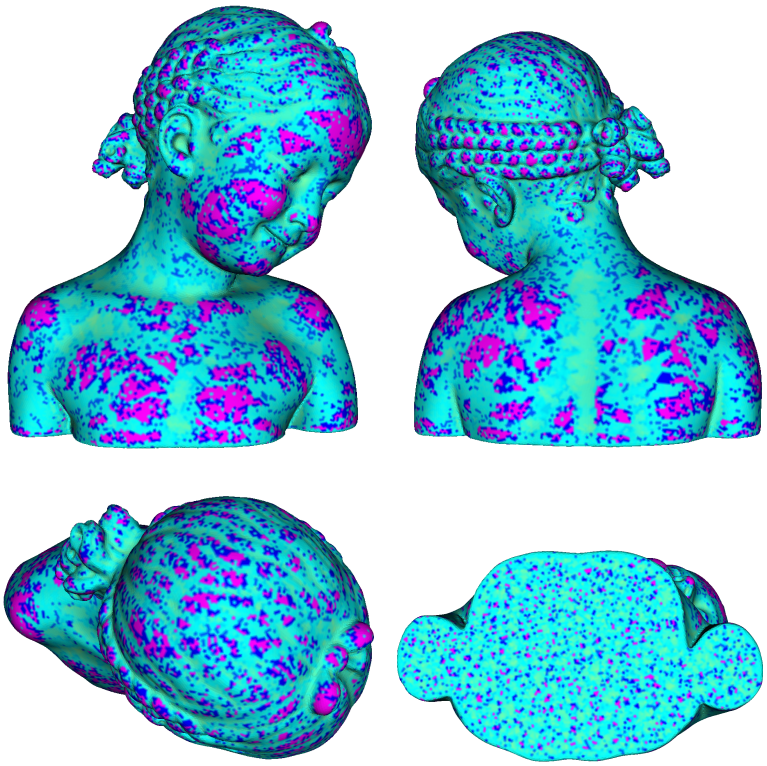
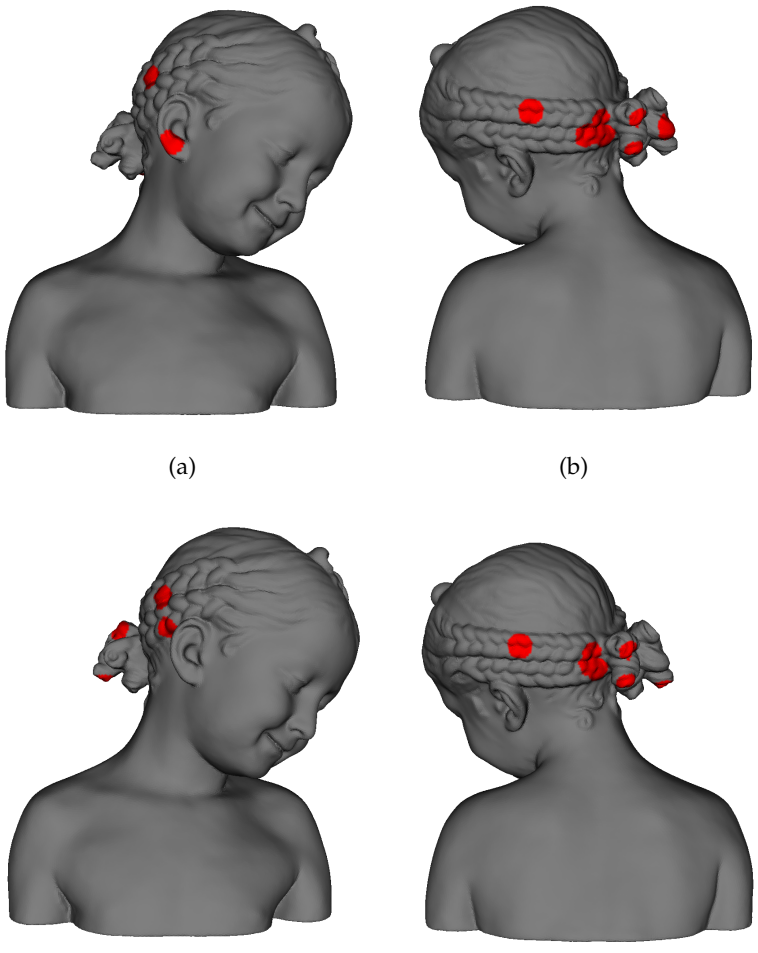


Figure 5-37: Vertices coloured using cluster membership in 16×16 SOM with features SC

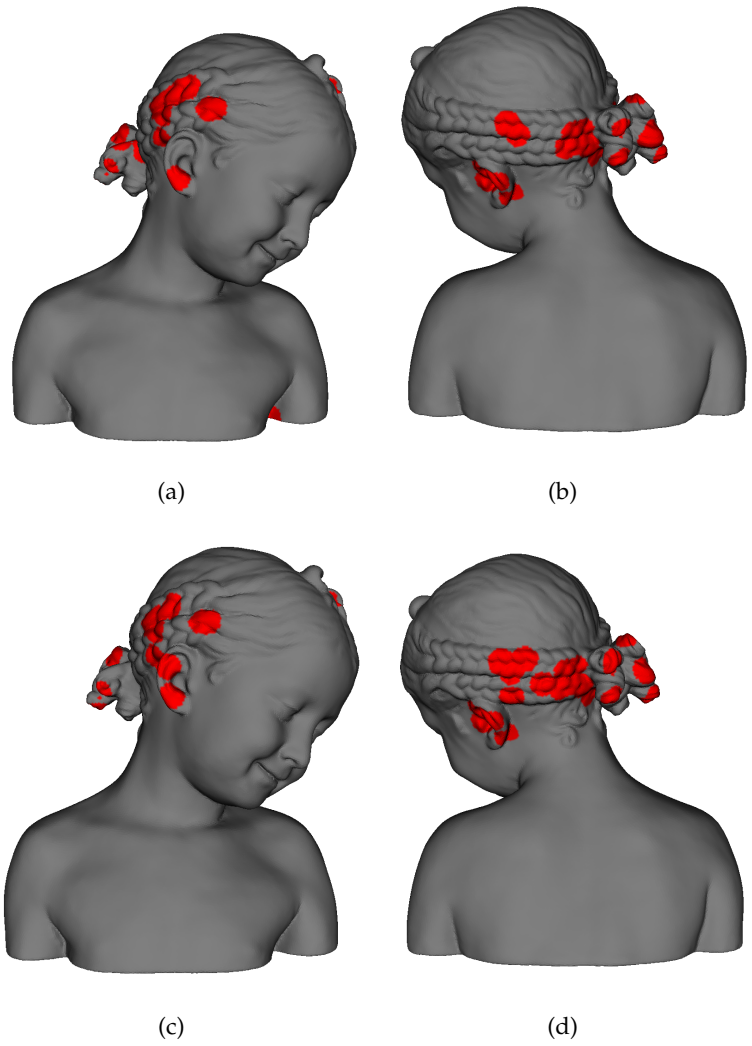


(a) (b)

(c) (d)

Figure 5-38: Top 10 Region matches using LMS in (a) and (b) and EMD in (c) and (d) with a 16×16 SOM and features SC

top 10 results after expanding the index (Figure 5-41) shows less regions returned from the search that are visually similar to the original search criteria. This didn't improve when inspecting the results of both the LMS and EMD distance measures with the top 50 results with results appearing on the nominally flat underside of the model. Although these regions are similar when comparing based on BMU distribution they aren't similar visually.



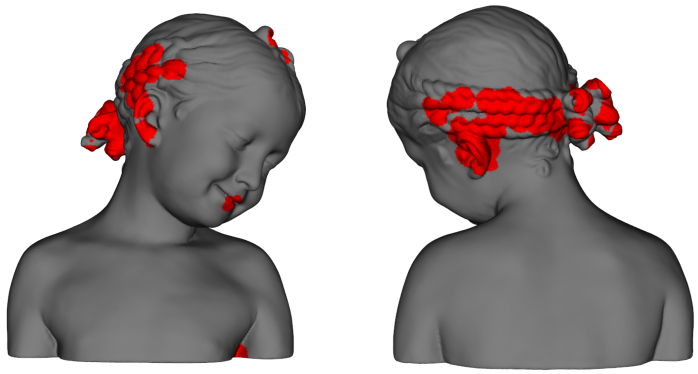
(a) (b)

(c) (d)

Figure 5-39: Top 50 Region matches using mean squared difference (a, b) and EMD (c, d) with a 16×16 SOM and features SC

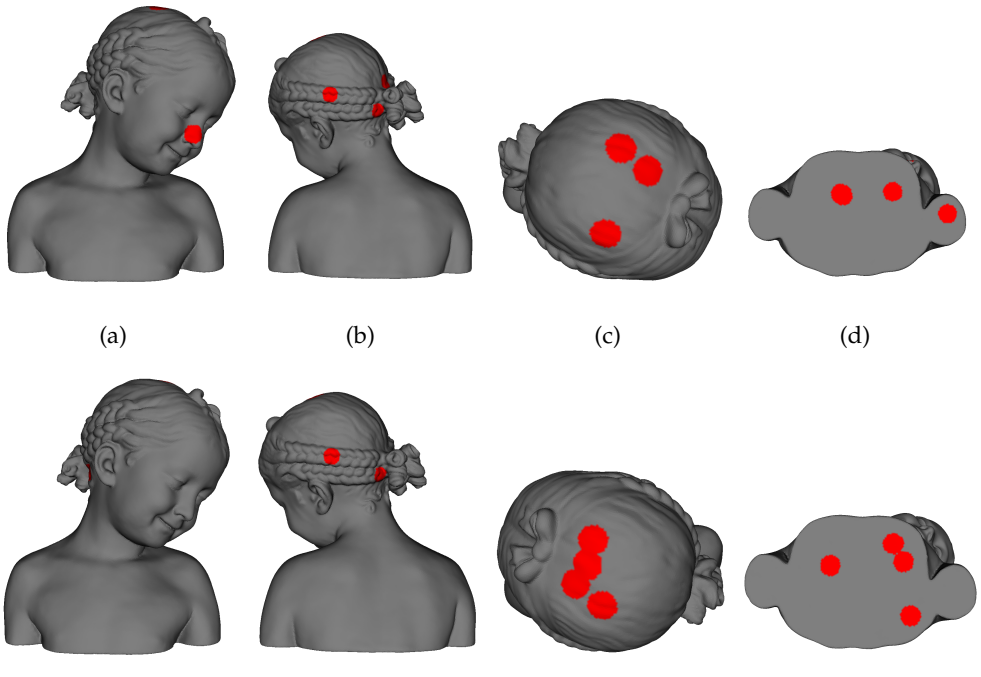
5.4 Object class identification

In this Section we will evaluate our SOM based descriptor using the Purdue CAD benchmark dataset [Jayanti et al., 2006]. We attempted to use the Princeton search dataset [Shilane et al., 2004] but encountered some difficulties due to poor quality, defective topologies and other problems with the models. Our technique relies on a reasonably uniform triangle size and distribution and surfaces without degeneracies such as vertices encoded



(a) (b)

Figure 5-40: Top 200 Region matches using EMD with a 16×16 SOM and features SC



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 5-41: Top 10 Region matches using LMS (a, b, c, d) and EMD (e, f, g, h) with a 16×16 SOM and features SC after adding entries with varied radius to the index

as being adjacent to themselves. Our assumptions worked well with input meshes generated from denoised range data and CAD models but isn't suited to every kind of 3D model unless they are pre-processed.

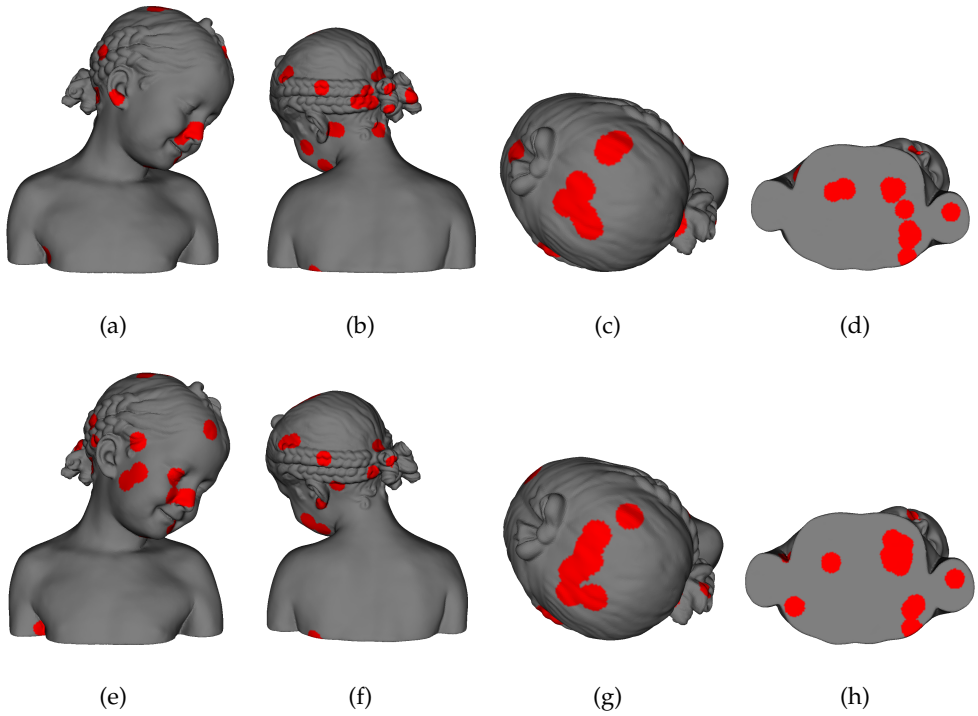


Figure 5-42: Top 50 Region matches using mean squared difference (a, b, c, d) and EMD (e, f, g, h) with a 16×16 SOM and features SC after adding entries with varying radii to the index

We uniformly remeshed the models in the Purdue dataset using technique described in Valette et al. [2008] with the implementation provided by the authors [S. Valette, 2008] with a fixed target vertex budget of 10,000 points to give a reasonable spread of vertices from which we calculated our descriptors. The models shown in Figure 5-21 on page 202 (cube and wedge) and Figure 3-1 on page 56 (paraboloid, hyperbolic paraboloid and paraboloid) were used to train a SOM using the parameters shown in Figure 5-25 on page 205: $KHSC$, 10×10 SOM and 100 coarse training epochs.

There are two object set groupings: high and low level. The high level categories are: Flat-Thin Walled Components, Rectangular-Cubic Prism and Solid of Revolution. The low level categories (43) were grouped into more categories such as Machined Plates, Thick Plates, Cylindrical Parts and Spoked Wheels to form a ground truth set. We followed the technique described by Jayanti et al. [2006] and calculated the mean BMU distribution

histogram for each of the object categories by taking the average of the SOM responses of every second model in that set.

The histogram of mean responses is used as the representative SOM response used to identify objects that belong to that class. The experimental results were obtained by finding the closest match between the SOM response histogram and the set of representative SOM response histogram for each of the object categories. The EMD distance metric was used to make the comparisons and the matches are then ranked by distance to the centre of the closest matching cluster. For each of the result we ranked them by distance to their object class SOM response histogram.

Our precision and recall results were obtained by comparing our results for each category against the ground truth in their sorted order. The precision and recall values are calculated within the interval of $[0, 1]$ as described by the TREC standard [Voorhees and Buckland, 2006]. Precision is the proportion of correct results returned $\frac{TruePositive}{TruePositive+FalsePositive}$ and recall is the proportion of total correct responses returned $\frac{TruePositive}{TruePositive+FalseNegative}$, the ideal result for both would be 1.0.

The precision-recall plots for our experiment (Figure 5-43) shows that for the high level categories our technique produced higher precision and recall compared to the object class recognition techniques presented by Jayanti et al. [2006] shown in Figures 5-44, 5-45 and 5-46. We then applied this technique to identify object subclasses contained within the CAD benchmark dataset and examined the precision and recall characteristics of the categories and found that in many cases our TREC standard precision and recall curves [Voorhees and Buckland, 2006] were comparable to those published in Jayanti et al. [2006]. Our method was better than the techniques evaluated in the CAD benchmark paper for some object categories such as the Spoked Wheels (Figure 5-47) and Contact Switches (Figure 5-48).

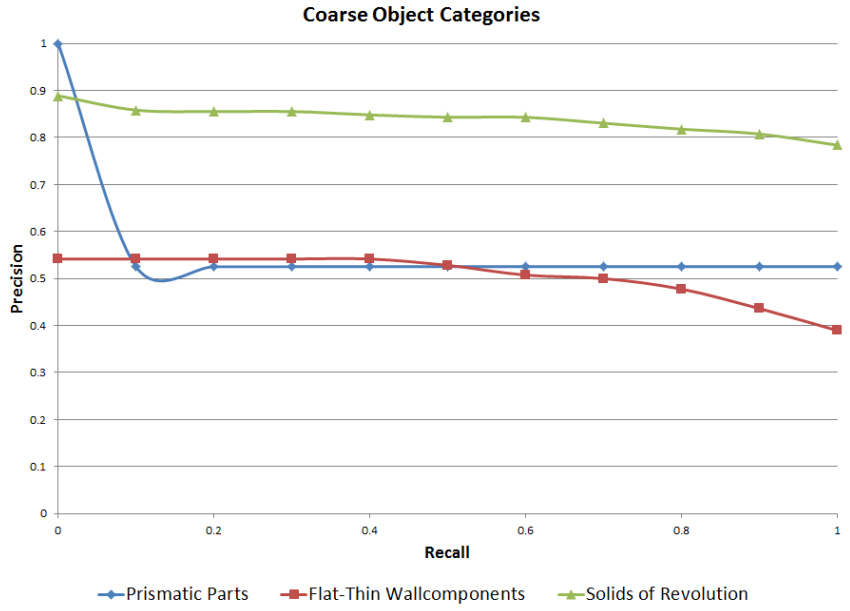


Figure 5-43: Precision Recall curve for Coarse Object Categories using SOM based descriptor

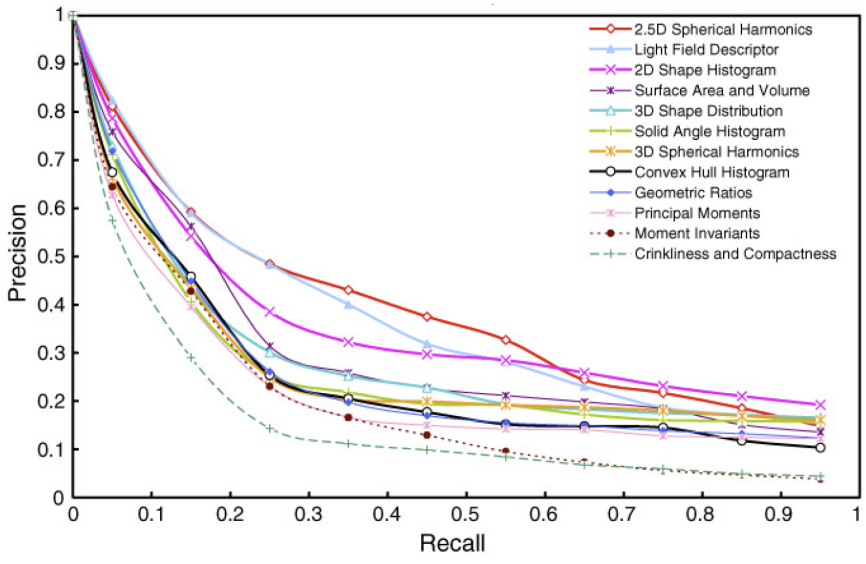


Figure 5-44: Precision Recall curve for for the Flat-Thin Walled Components surface set from Jayanti et al. [2006]

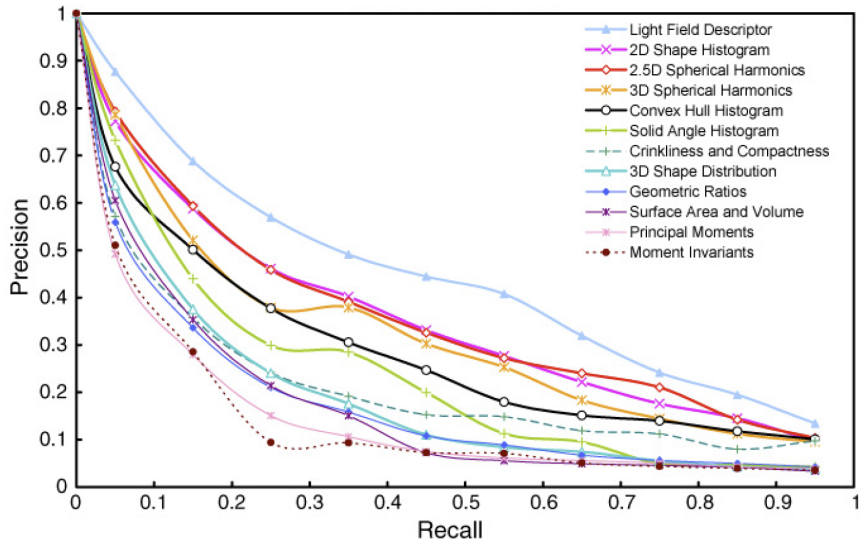


Figure 5-45: Precision Recall curve for the Solids of Revolution surface set from Jayanti et al. [2006]

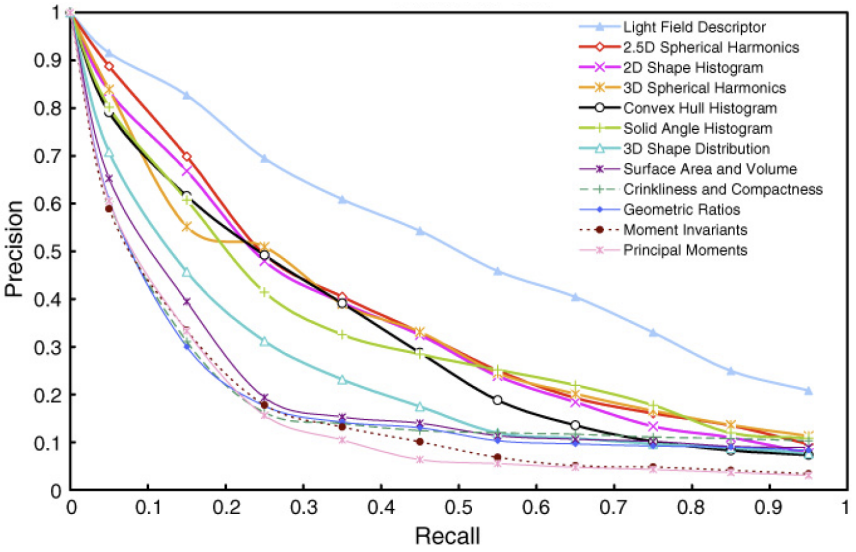
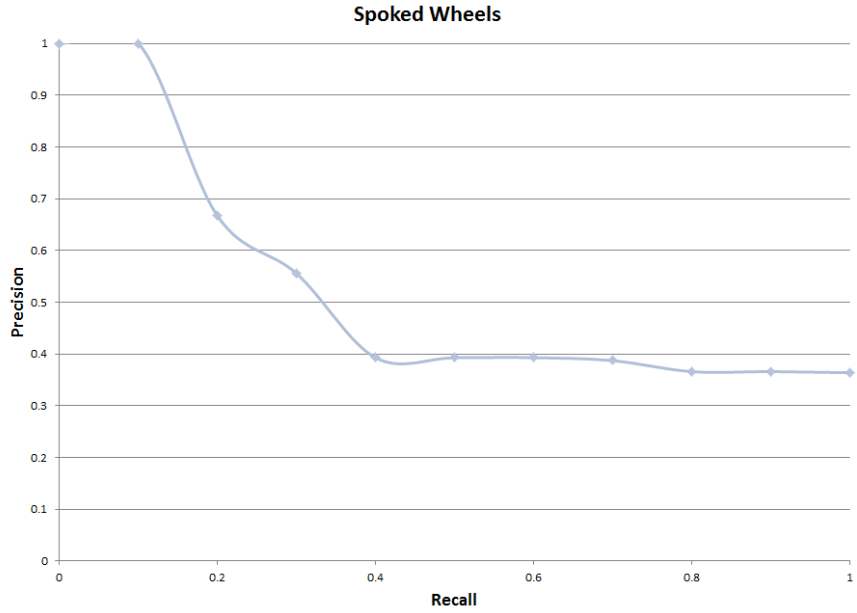


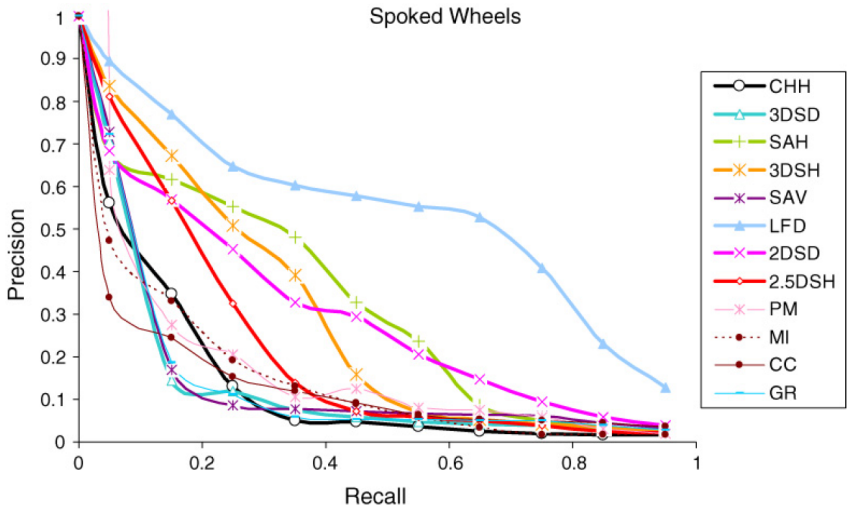
Figure 5-46: Precision Recall curve for the Prismatic Parts surface set from Jayanti et al. [2006]

5.5 Conclusion

In this Chapter we demonstrated how distributions of cluster representation within a sphere bound region or whole surfaces could be used to build an index and retrieve sim-



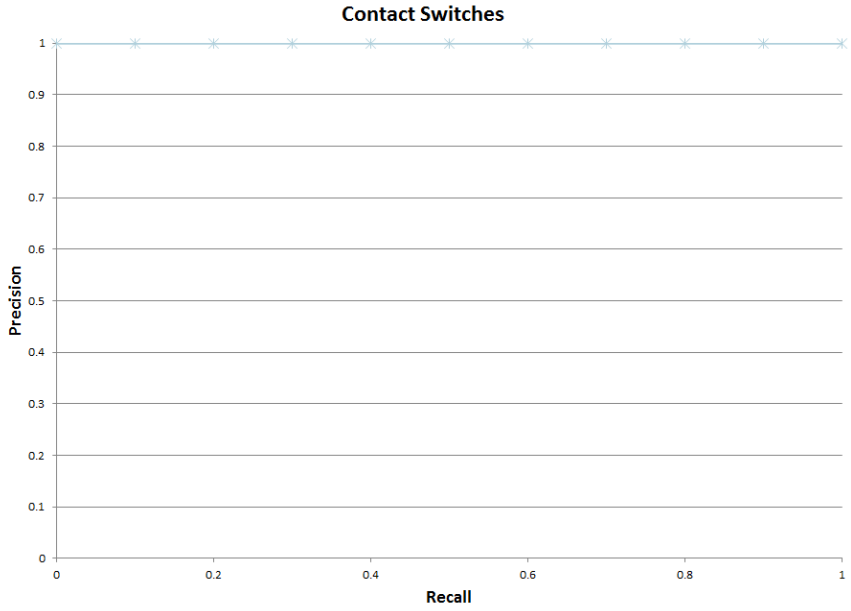
(a) BMU distribution



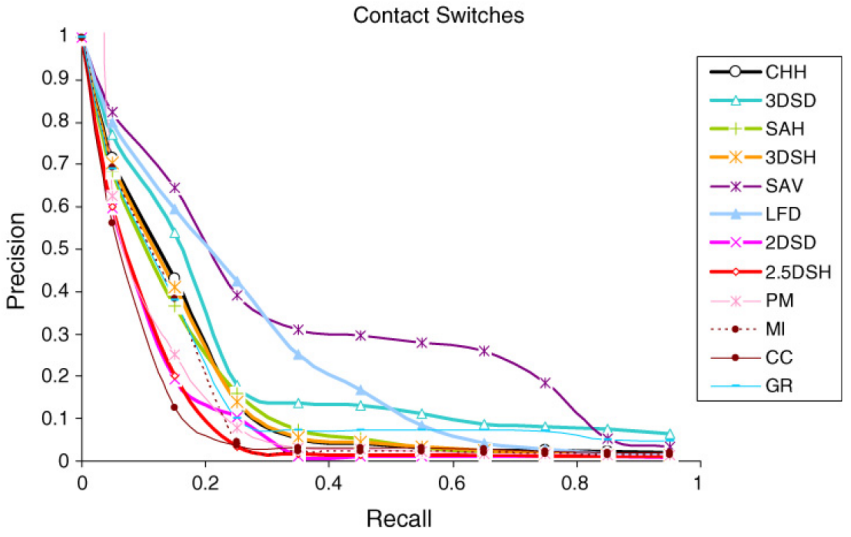
(b) Other techniques from Jayanti et al. [2006]

Figure 5-47: Precision Recall curve for the Spoked Wheels object set

ilar regions using a SOM trained using exemplar surfaces of the object being indexed. The technique worked well when matching regions that share the same neighbourhood sphere radius, low levels of noise in the mesh being indexed and the distributions themselves were interesting. We evaluated three different sources of training data: the objects



(a) BMU distribution



(b) Other techniques from Jayanti et al. [2006]

Figure 5-48: Precision Recall curve for the Contact Switch object set

being indexed, exemplar objects and the search region itself and found that the set of exemplar objects gave us the most consistent results compared to the other two options. This is interesting as we observed noisy results in the previous sections when using this training data set to segment objects. The features HC gave us the best results for the

CAD model dataset as they were able to represent a broad range of surface types. The SOM dimensions of 16×16 gave us 256 possible region types, more than the 3×3 offered by Besl [1988] and Koenderink [1990], allowing the model to express a greater number of surface types in terms of their curvature.

By reducing the descriptor of a region to a bag of words we were able to focus the computationally intensive EMD and mean squared difference distance metrics on a subset of the possible matches by using a SQL query that extracted regions that shared a similar proportion of the dominant cluster in the search criteria. We were able to identify high and low level object classes in the CAD model benchmark with precision and recall results similar to that of other techniques in [Jayanti et al., 2006].

The effectiveness of our technique is dictated by the level of noise in the 3D objects being indexed, meshing parameters when converting from NURBS based models to regular triangulated surfaces, the training data and parameters used when building a SOM to be used at the classifier and a sufficient amount of variation in curvature to form distinctive region distributions. The EMD metric was better at matching visually similar regions as it took into account distances between clusters instead of focussing on each part of the distribution as a discrete entity. Our technique performs reasonably well when the region of interest contains enough variation in curvature to be interesting. When we selected relatively simple components such as a corner our method did not perform well but but once we introduced an object with a lot of variation in curvature we could find good matches.

Further research is needed to extend this technique by determining how to compliment the bag of words approach to describe regions with other techniques to encode spatial relationships between regions that are considered statistically unusual when compared to the population of regions in the index. Geometric hashing is one technique that could be used to solve this problem [Mokhtarian et al., 2001, Bebis et al., 1998, Mian et al., 2006].

Chapter 6

Conclusion

6.1 Conclusions

We investigated the use of Self-Organizing maps combined with vertex based curvature metrics to describe the vertices of free-form surfaces and applied this knowledge to segmentation and indexing and retrieval of these free-form surfaces.

As a part of this investigation we developed a complete experimental pipeline using our own SOM and curvature calculation implementations in Java. The experimental results, indexes, free-form surfaces and SOMs were all stored in a SQL database by using annotated code processed by OpenJPA at runtime. Our 3D models were visualized using our own code combined with the GUI widgets provided by VTK. Conversion between different file formats and from NURBS representations to OOGL was achieved using a combination of our own code, MeshLab, GiD, VTK and GMSH. SOM visualizations were obtained using our experimental output and the SOM toolkit implementation in Matlab.

We evaluated the effect of feature selection, training iterations, SOM size and training data set had on the classification of vertices from triangulated free-form surfaces. The training data sets used to evaluate the SOM were a set of paraboloid exemplars and the free-form surfaces being classified. When using the paraboloid training data set we found that the Q error stabilised and the percentage of neurons fired decreased as the SOM size increased. The Q error converged on different values depending upon feature selection. The regular behaviour was for the Q error and percentage of all neurons fired during classification to decay at a constant rate as the SOM size increased. When using $S\ C\ k_1$ k_2 and $H\ S$ feature sets the percentage of neurons fired remained high and using $K\ M$ and $K\ S$ the percentage of neurons fired was consistently low. In most cases increasing the number of coarse and fine training iterations decreased the Q error and increased the percentage of neurons fired during classification. We found that a 16×16 SOM using the features SC and the largest number of coarse and fine training iterations and the object being classified used as the training data set produced a SOM that yielded the lowest Q error. Very low values for Q were not always desirable is it usually indicated cases where the data was overfitted which introduced unwanted artefacts in the classification process.

The surface being processed was the best training data set because its distribution of vertex types perfectly matched the data being classified.

Our vertex classification was then successfully applied to the problem of segmenting free-form surfaces. The features $H\ C$ were found to be the most stable across all of the training data sets, which made sense as they have a smoothing effect on extreme values of k_1 and k_2. A low mean Q error after classifying the vertices of the surface being classified resulted in a noisy over segmented result. Well segmented surfaces were achieved by using no fine training iterations and 10 training epochs to ensure the SOM wasn't too sensitive to small variations in the vertices curvature properties. The best segmentation results consistently came from SOMs trained using the surface vertex properties as the training data set. The paraboloid data set performed poorly as its vertex type distribution was very different to the surface being segmented. A set of random surfaces was also used to evaluate the impact of training data selection. The random surfaces performed better than the paraboloids when classifying the CAD part, firing a greater number of unique neurons, as there was more variation in curvature compared to the paraboloids.

In the final chapter we applied our techniques to the problem of indexing and retrieval of free-form surfaces, examining partial and whole surface matching and object class recognition. We took a bag of words approach by describing regions, whole surfaces and object classes as BMU distributions. We were able to find similar sphere bound local regions on a single surface using that surface as training data and a least mean squared distance between BMU distributions as our distance metric. This worked well for low noise surfaces with matches sharing the same sphere radius. We evaluated our technique using three training data sets: the surfaces being indexed, search criteria and exemplar paraboloids. The paraboloids and features HC gave us the most consistent results. Our descriptor was able to identify some matching regions on noisy surfaces, but didn't perform nearly as well compared to matching on low-noise surfaces. The technique performed poorly when the surfaces showed little variation in curvature and varied in scale (Section 5.2.) Our attempts to address the problem by using a greater range of sphere radii when building our index didn't improve the results as we did not encode spatial relationships.

Since our regions are represented as BMU distributions we could reduce the number of records to process when searching for matches by using a SQL query against our database of distributions to return a subset of results and then apply the computationally intensive, but more effective, earth movers distance (EMD) metric. The EMD as a distance metric suited our BMU distribution representation because it takes into account the feature distance between the histogram buckets instead of each histogram value in isolation. The EMD gave us more relevant results compared to the least mean squared difference metric. We then used our descriptors to build representations of object classes from a CAD benchmark dataset [Jayanti et al., 2006] and achieved recognition results comparable with other modern surface descriptors. The EMD implementation was created by Pele and Werman [2009] running under Matlab, we accessed the algorithm from our Java code using the Matlab to Java bridge JAMAL [Khadkevich, 2013].

During this investigation we identified the strengths and weaknesses of our approach. To improve the quality of our existing curvature based surface features we need to preprocess noisy data sets using noise reduction techniques such as surface fitting, smoothing and statistical methods. These techniques will also help address the problems of making comparisons between objects meshed with different parameters and varying levels of noise. The inclusion of features such as dihedral angle and directions of curvature and other spatial relationship descriptors make the classifier sensitive to spatial properties, further improving its descriptive qualities. These vertex descriptors could be augmented with topological descriptors such as Reeb and spectral graphs. Complimentary methods to the SOM such as level set methods, expectation-maximization and other related algorithms could be used to reduce the reliance on selecting the correct SOM size parameter (we just used square SOMs).

The exemplar surfaces used in this thesis were chosen as they represented a broad range of curve types but more investigations are required to identify better sets of exemplars and how to represent these surfaces so we can reduce or eliminate the dependence on the meshing parameters when converting these surfaces to meshes. We observed that noise present when classifying the exemplar surfaces such as the hyperbolic paraboloid

tended to affect the results when classifying the vertices of other surfaces using a SOM trained with the exemplars. One possible way to avoid this problem is to change how we describe the surfaces in our experiments and even move away from mesh representations entirely.

We were unable to use our best classifier results, achieved using the data being classified as the training data, when finding matching regions across different surfaces because it is difficult to compare BMU distributions from different SOMs. An efficient technique making these cross-SOM comparisons possible could yield better results. It is possible to use the EMD, but then we wouldn't be able to query our index using a SQL query to reduce the search space.

Bibliography

M. A. Abidi, S. Yasuki, and P. B. Crilly. Image compression using hybrid neural networks. *Digest of Technical Papers—IEEE International Conference on Consumer Electronics*, pages 70–71, 1994.

Matti Aksela, Ramunas Girdziusas, Jorma Laaksonen, Erkki Oja, and Jari Kangas. Methods for adaptive combination of classifiers with application to recognition of handwritten characters. *IJDAR*, 6(1):23–41, 2003.

A. Alrashdan, S. Motavalli, and B. Fallahi. Automatic segmentation of digitized data for reverse engineering applications. *IIE Transactions*, 32:59–69, 2000.

Nina Amenta, Stuart Levy, Tamara Munzner, and Mark Phillips. Geomview: a system for geometric visualization. In *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*, pages 412–413, New York, NY, USA, 1995. ACM Press. ISBN 0-89791-724-3. doi: http://doi.acm.org/10.1145/220279.220327.

Mihael Ankerst, Gabi Kastenmüller, Hans-Peter Kriegel, and Thomas Seidl. 3d shape histograms for similarity search and classification in spatial databases. In *Proceedings of the 6th International Symposium on Advances in Spatial Databases*, SSD '99, pages 207–226, London, UK, UK, 1999. Springer-Verlag. ISBN 3-540-66247-2. URL `http://dl.acm.org/citation.cfm?id=647226.719092`. [Online; accessed 8-Aug-2013].

Nicholas Apostoloff and Andrew W. Fitzgibbon. Automatic video segmentation using spatiotemporal t-junctions. In Mike J. Chantler, Robert B. Fisher, and Emanuele Trucco, editors, *BMVC*, pages 1089–1098. British Machine Vision Association, 2006. ISBN 1-904410-14-6.

Artec Group inc. 3D models, 2012. URL http://www.artec3d.com/tag/?tags=Artec+MH. [Online; accessed 4-Apr-2013] Creative Commons License.

J. Assfalg, M. Bertini, A.D. Bimbo, and P. Pala. Content-based retrieval of 3-d objects using spin image signatures. *Multimedia, IEEE Transactions on*, 9(3):589 –599, april 2007. ISSN 1520-9210. doi: 10.1109/TMM.2006.886271.

U.S. Product Data Association. Initial Graphics Exchange Specification 5.3, 1999. URL http://www.iges5x.org/archives/version5x/. [Online; accessed 8-Aug-2013].

Mohammed Attik, Laurent Bougrain, and Frédéric Alexandre. Self-organizing map initialization. In Wlodzislaw Duch, Janusz Kacprzyk, Erkki Oja, and Slawomir Zadrozny, editors, *ICANN (1)*, volume 3696 of *Lecture Notes in Computer Science*, pages 357–362. Springer, 2005. ISBN 3-540-28752-3.

J. M. Barbalho, A. D. D. Neto, J. A. E. Costa, and M. L. A. Netto. Hierarchical SOM applied to image compression. In *Proceedings of the International Joint Conference on Neural Networks*, volume 1, pages 442–447. Univ. Federal do Rio Grande do Norte, Dept. of Electrical Engineering, Lab. of Computer Eng. and Automation, 2001.

R. A. Beard and K. S. Rattan. A neural network system for robot vision. In *Proc. NAECON 1989, IEEE 1989 National Aerospace and Electronics Conference*, volume IV, pages 1920–1921, Piscataway, NJ, 1989. IEEE Service Center.

George Bebis, Michael Georgiopoulos, and Niels da Vitoria Lobo. Using self-organizing maps to learn geometric hash functions for model-based object recognition. *IEEE Transactions on Neural Networks*, 9:560–570, 1998.

Rachid Benlamri and Yousuf Al-Marzooqi. Free-form object segmentation and representation from registered range and color images. *Image Vision Comput.*, 22(9):703–717, 2004.

P. J. Besl and R. C. Jain. Segmentation through variable-order surface fitting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(2):167–192, 1988. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/34.3881.

Paul J. Besl. *Surfaces in Range Image Understanding*. Springer Verlag, New York, 1988.

S.M. Bhandarker, J. Koh, and M. Suk. Multiscale image segmentation using a hierarchical self organizing map. *Neurocomputing*, 14(3):241–272, 1997.

Silvia Biasotti. Reeb graph representation of surfaces with boundary. In *Proceedings of the Shape Modeling International 2004*, pages 371–374, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2075-8. URL `http://dl.acm.org/citation.cfm?id=998687.1007069`. [Online; accessed 8-Aug-2013].

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:2003, 2002. URL `http://www.cs.princeton.edu/~blei/papers/BleiNgJordan2003.pdf`. [Online; accessed 8-Aug-2013].

S. K. Bose, K. K. Biswas, and S. K. Gupta. An integrated approach for range image segmentation and representation. *Artificial Intelligence in Engineering*, 10(3):243–252, August 1996. doi: 10.1016/0954-1810(95)00035-6. URL `http://dx.doi.org/10.1016/0954-1810(95)00035-6`. [Online; accessed 8-Aug-2013].

T. Botterill, R. Green, and S. Mills. A bag-of-words speedometer for single camera slam. In *Image and Vision Computing New Zealand*, pages 91–96, 2009.

Antonio Chella and Roberto Pirrone. A two stage neural architecture for segmentation and superquadrics recovery from range data. In Maria Marinaro and Roberto Tagliaferri, editors, *Neural Nets, 13th Italian Workshop on Neural Nets, WIRN VIETRI 2002, Vietri sul Mare, Italy, May 30-June 1, 2002, Revised Papers*, volume 2486 of *Lecture Notes in Computer Science*, pages 132–139. Springer, 2002.

CIMME. GiD Pre and Post Processor, 2007. URL `http://www.gid.cimne.upc.edu/`. [Online; accessed 8-Aug-2013].

J. Clark. Object digitization for everyone. *Computer*, 44(10):81 –83, oct. 2011. ISSN 0018-9162. doi: 10.1109/MC.2011.320.

S.A. Coleman, B.W. Scotney, and S. Suganthan. Edge detecting for range data using laplacian operators. *Image Processing, IEEE Transactions on*, 19(11):2814 –2824, nov. 2010. ISSN 1057-7149. doi: 10.1109/TIP.2010.2050733.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005. URL `http://lear.inrialpes.fr/pubs/2005/DT05`. [Online; accessed 8-Aug-2013].

C. Dorai and A.K. Jain. Cosmos-a representation scheme for 3d free-form surfaces. In *Proceedings of the 5th International Conference on Computer Vision*, pages 1024–1029, Boston, USA, 1995.

C. Dorai and A.K. Jain. Cosmos-a representation scheme for 3d free-form objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1115–1130, 1997a.

C. Dorai and A.K. Jain. Shape spectra based view grouping and matching of 3d free-form objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1139–1145, 1997b.

Susan T. Dumais. Latent semantic analysis. *Annual Review of Information Science and Technology*, 38(1):188–230, 2004. ISSN 1550-8382. doi: 10.1002/aris.1440380105. URL `http://dx.doi.org/10.1002/aris.1440380105`. [Online; accessed 8-Aug-2013].

Robert B. Fisher Emanuele Trucco. Experiments in curvature-based segmentation of range data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(2):177–182, 1995.

Bianca Falcidieno. AIM@SHAPE shape repository, 2009. URL `http://shapes.aim-at-shape.net/`. [Online; accessed 8-Aug-2013].

Bianca Falcidieno, Michela Spagnuolo, Marios Pitikakis, George Vasilakis, Alejandra García-Rojas, and Laura Papaleo. Aim@shape: Research advantages and future contributions. In Yannis S. Avrithis, Yiannis Kompatsiaris, Steffen Staab, and Noel E. O'Connor, editors, *SAMT (Posters and Demos)*, volume 233 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

T. J. Fan, G. Medioni, and R. Nevatia. Recognizing 3-d objects using surface descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1140–1157, 1989.

Apache Foundation. Apache commons, 2011a. URL http://commons.apache.org. [Online; accessed 8-Aug-2013].

Apache Foundation. Derby, 2011b. URL http://db.apache.org/derby. [Online; accessed 8-Aug-2013].

Apache Foundation. OpenJPA, 2011c. URL http://openjpa.apache.org. [Online; accessed 8-Aug-2013].

Jing Fu, Sanjay B. Joshi, and Timothy W. Simpson. Shape differentiation of freeform surfaces using a similarity measure based on an integral of gaussian curvature. *Computer-Aided Design*, 40(3):311 – 323, 2008. ISSN 0010-4485. doi: 10.1016/j.cad.2007.11.006. URL http://www.sciencedirect.com/science/article/pii/S0010448507002461. [Online; accessed 8-Aug-2013].

T. Funkhouser and P. Shilane. Partial matching of 3d shapes with priority-driven search. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, pages 131–142, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association. ISBN 3-905673-36-3. URL http://dl.acm.org/citation.cfm?id=1281957.1281974. [Online; accessed 8-Aug-2013].

Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs. A search engine for 3d models. *ACM Transactions on Grahics*, 22(1):83–105, 2003.

Eric Gaba. View of the planes establishing the main curvatures on a minimal surface, 2006. URL http://en.wikipedia.org/wiki/File:Minimal_surface_curvature_planes-en.svg. [Online; accessed 12-Dec-2012] GNU Free Documentation License.

Ran Gal and Daniel Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Trans. Graph.*, 25:130–150, January 2006. ISSN 0730-0301. URL http://doi.acm.org/10.1145/1122501.1122507. [Online; accessed 8-Aug-2013].

Yue Gao, You Yang, Qionghai Dai, and Naiyao Zhang. 3d object retrieval with bag-

of-region-words. In *ACM Multimedia Conference*, pages 955–958, 2010. doi: 10.1145/1873951.1874122.

Timothy Gatzke, Cindy Grimm, Michael Garland, and Steve Zelinka. Curvature maps for local shape comparison. In *Proceedings of the International Conference on Shape Modeling and Applications 2005*, pages 246–255, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2379-X. doi: 10.1109/SMI.2005.13. URL `http://dl.acm.org/citation.cfm?id=1097876.1098476`. [Online; accessed 8-Aug-2013].

Christophe Geuzaine and Jean-Francois Remacle. *Gmsh*. http://www.geuz.org/gmsh, 1.12 edition, August 2003.

Alfred Gray. *Modern Differential Geometry of Curves and Surfaces with Mathematica*. CRC Press, Inc., Boca Raton, FL, USA, 1998. ISBN 0849371643.

Michel Haritopoulos, Hujun Yin, and Nigel M. Allinson. Image denoising using self-organizing map-based nonlinear independent component analysis. *Neural Networks*, 15(89):1085 – 1098, 2002. ISSN 0893-6080. URL `http://www.sciencedirect.com/science/article/pii/S0893608002000813`. [Online; accessed 8-Aug-2013].

S. Haykn. *Neural Networks: A Comprehensive Foundation*. Maxwell Macmillan International, Sydney, 1994. ISBN 0-02-352761-7.

Masaki Hilaga, Yoshihisa Shinigawa, Taku Kohmura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *SIGGRAPH 2001*, pages 203–212. ACM Press, 2001.

Berthold K. P. Horn. Extended Gaussian images. A.I. Memo 740, Massachusetts Institute of Technology - Artifical Intelligence Laboratory, July, 1983 1983.

Short Tutorial: Recognizing and Learning Object Categories, Kyoto, Japan, 2009. IEEE Computer Society. URL `http://people.csail.mit.edu/torralba/shortCourseRLOC/index.html`. [Online; accessed 8-Aug-2013].

Natraj Iyer, Subramaniam Jayanti, Kuiyang Lou, Yagnanarayanan Kalyanaraman, and Karthik Ramani. Three-dimensional shape searching: state-of-the-art review and future trends. *Computer-Aided Design*, 37(5):509, 2005.

Subramaniam Jayanti, Yagnanarayanan Kalyanaraman, Natraj Iyer, and Karthik Ramani. Developing an engineering shape benchmark for cad models. *Computer-Aided Design*, 38(9):939 – 953, 2006. ISSN 0010-4485. URL http://www.sciencedirect.com/science/article/pii/S001044850600100X. [Online; accessed 8-Aug-2013].

Xiaoyi Jiang and Horst Bunke. Robust edge detection in range images based on scan line approximation. *Computer Vision and Image Understanding*, 73:183–199, 1996.

A.E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433 –449, may 1999. ISSN 0162-8828. doi: 10.1109/34.765655.

L.P. Kaelbling, M.L. Littman, and Andrew Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

Sagi Katz and Ayellet Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Trans. Graph.*, 22(3):954–961, 2003. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/882262.882369.

Michael Misha Kazhdan. Extended Gaussian images, 2004. URL http://www.cs.jhu.edu/~misha/Fall04/EGI1.ppt. [Online; accessed 8-Aug-2013].

Maksim Khadkevich. JAMAL - JAva MAtlab Linking, 2013. URL http://jamal.khadkevich.org/documentation.html. [Online; accessed 8-Aug-2013].

I. Khalifa, M. Moussa, and M. Kamel. Range image segmentation with application to cad model acquisition. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, volume 2, pages 740–743 vol.2, 2000. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=899815. [Online; accessed 8-Aug-2013].

Melody Y. Kiang, Uday R. Kulkarni, Michael Goul, Robert T. H. Chi, Efraim Turban, and Andrew Philippakis. Improving the effectiveness of self-organizing map networks using a circular kohonen layer. In *Hawaii International Conference on System Sciences*, pages 521–529, 1997.

J. J. Koenderink and J. J. Van Doorn. Surface shape and curvature scales. *Image and Vision Computing*, 10(8):557–565, 1992.

Jan J Koenderink. *Solid shape*. MIT Press, Cambridge, Mass., 1990.

J. Koh, M. Suk, and S. M. Bhandarkar. A multi-layer kohonen's self-organizing feature map for range image segmentation. In *Neural Networks, 1993., IEEE International Conference on*, pages 1270–1275 vol.3, 1993. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=298740`. [Online; accessed 8-Aug-2013].

J. Koh, M. Suk, and S.M. Bhandarkar. A multilayer self-organizing feature map for range image segmentation. *Neural Networks*, 8(1):67–86, 1995.

T. Kohonen. *Self-Organizing-Maps*. Springer-Verlag, Berlin, third edition, 2001. ISBN 3-540-67921-9.

Farid Abedan Kondori, Shahrouz Yousefi, Haibo Li, Samuel Sonning, and Sabina Sonning. 3d head pose estimation using the kinect. In *Wireless Communications and Signal Processing (WCSP), 2011 International Conference on*, pages 1 –4, nov. 2011. doi: 10.1109/WCSP.2011.6096866.

Yu-Kun Lai, Shi-Min Hu, Ralph R. Martin, and Paul L. Rosin. Rapid and effective segmentation of 3d models using random walks. *Comput. Aided Geom. Des.*, 26:665–679, August 2009. ISSN 0167-8396. URL `http://dl.acm.org/citation.cfm?id=1542572.1543081`. [Online; accessed 8-Aug-2013].

Rong Liu and Hao Zhang. Segmentation of 3d meshes through spectral clustering. In *Pacific Conference on Computer Graphics and Applications*, pages 298–305. IEEE Computer Society, 2004. ISBN 0-7695-2234-3.

Xiuwen Liu and DeLiang L. Wang. Range image segmentation using a relaxation oscillator network. *IEEE Transactions on Neural Networks*, 10(3):564–573, May 1999.

D. Heckerman M. Sahami, S. Dumais and E. Horvitz. A bayesian approach to filtering junk e-mail. In *AAAI'98 Workshop on Learning for Text Categorization*. The AAAI Press, 1998. ISBN 978-1-57735-418-5. URL `http://research.microsoft.com/en-us/um/people/horvitz/junkfilter.htm`. [Online; accessed 8-Aug-2013].

A. D. MacLennan and G. A. W. West. Region matching in free-form surfaces using self organizing maps. In *DICTA*, pages 578–585. IEEE Computer Society, 2008.

A.D. MacLennan, G.A.W West, and M. Cardew-Hall. Investigating the segmentation of freeform triangulated surfaces using a self-organizing map. In *Proceedings of 2006 ASME International Design Engineering Technical Conferences and the Computers and Information in Engineering Conference*, Philadelphia, PA, U.S.A., September 2006.

A.P. Mangan and R.T. Whitaker. Partitioning 3d surface meshes using watershed segmentation. *Visualization and Computer Graphics, IEEE Transactions on*, 5(4):308–321, 1999.

Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In Hans-Christian Hege and Konrad Polthier, editors, *Visualization and Mathematics III*, pages 35–54, Berlin, Germany, 2002. Springer. ISBN 978-3-540-01295-5.

M.A.A. Mhamdi, D. Ziou, A. Lachkar, and S. El Alaoui Ouatik. An efficient method for 3d objects retrieval based on fixed number of 2d views approach. In *I/V Communications and Mobile Network (ISVC), 2010 5th International Symposium on*, pages 1 –4, 30 2010-oct. 2 2010. doi: 10.1109/ISVC.2010.5654898.

Ajmal S. Mian, Mohammed Bennamoun, and Robyn Owens. Three-dimensional model-based object recognition and segmentation in cluttered scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28:1584–1601, October 2006. ISSN 0162-8828. URL `http://dx.doi.org/10.1109/TPAMI.2006.213`. [Online; accessed 8-Aug-2013].

Microsoft Corporation, 2011. URL `http://www.kinectforwindows.org/`. [Online; accessed 8-Aug-2013].

Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 1969. ISBN 0262630222.

F. Mokhtarian, N. Khalili, and P. Yuen. Multi-scale free-form 3d object recognition using 3d models. *Image and Vision Computing*, 19(5):271–281, 2001.

A.B. Moreno and A.Sanchez. GavabDB: A 3d face database. In C. Garcia et al, editor, *Proc. 2nd COST Workshop on Biometrics on the Internet: Fundamentals, Advances and Applications*, pages 77–82. Ed. Univ. Vigo, 2004. URL `http://www.gavab.es/recursos_en.html#GavabDB`. [Online; accessed 13-Aug-2013].

B.C. Munsell, P. Dalal, and Song Wang. Evaluating shape correspondence for statistical shape analysis: A benchmark study. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(11):2023 –2039, nov. 2008. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.70841.

New Dimension Systems. 3D Content Central, 2011. URL `http://www.3dcontentcentral.com/parts/supplier/New-Dimension-Systems.aspx`. [Online; accessed 8-Aug-2013].

R. Ohbuchi, M. Tezuka, T. Furuya, and T. Oyobe. Squeezing bag-of-features for scalable and semantic 3d model retrieval. In *Content-Based Multimedia Indexing (CBMI), 2010 International Workshop on*, pages 1 –6, june 2010. doi: 10.1109/CBMI.2010.5529890.

Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Transactions on Grahics*, 21(4):807–832, 2002.

Libella Oy, Som Toolbox For Matlab, Juha Vesanto, Juha Vesanto, Johan Himberg, Johan Himberg, Esa Alhoniemi, Esa Alhoniemi, Juha Parhankangas, and Juha Parhankangas. Som toolbox for matlab 5, 2000. URL `http://www.cis.hut.fi/projects/somtoolbox/`. [Online; accessed 8-Aug-2013].

Xian Pan, Xiuzi Ye, and Sanyuan Zhang. 3d mesh segmentation using a two-stage merging strategy. In *Proceedings of the Fourth International Conference on Computer and Information Technology (CIT'04)*, pages 730–733, September 2004. doi: 10.1109/CIT.2004.1357281.

Patricio Parada and Javier Ruiz del Solar. Texture synthesis using image pyramids and self-organizing maps. *Image Analysis and Processing, International Conference on*, 0:0244, 2001. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=957016&isnumber=20665`. [Online; accessed 8-Aug-2013].

In Kyu Park, Kyoung Mu Lee, and Sang Uk Lee. Models and algorithms for efficient multiresolution topology estimation of measured 3-d range data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 33(4):706–711, 2003.

Valerio Pascucci, Giorgio Scorzelli, Peer-Timo Bremer, and Ajith Mascarenhas. Robust on-line computation of reeb graphs: simplicity and speed. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM. URL http://doi.acm.org/10.1145/1275808.1276449. [Online; accessed 8-Aug-2013].

Ofir Pele and Michael Werman. A linear time histogram metric for improved sift matching. In *ECCV*, 2008.

Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *ICCV*, 2009.

S. Peleg, M. Werman, and H. Rom. A unified approach to the change of resolution: space and gray-level. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(7):739 – 742, jul 1989. ISSN 0162-8828. doi: 10.1109/34.192468.

R. Petre, T. Zaharia, and F. Preteux. An experimental evaluation of view-based 2d/3d indexing methods. In *Electrical and Electronics Engineers in Israel (IEEEI), 2010 IEEE 26th Convention of*, pages 000924 –000928, nov. 2010. doi: 10.1109/EEEI.2010.5661944.

M. Pöllä, T. Honkela, M. Oja, S. Kaski, and T. Kohonen. Bibliography of self-organizing map (som) papers, 2008. URL http://www.cis.hut.fi/research/som-bibl/. [Online; accessed 8-Aug-2013].

Anshuman Razdan and MyungSoo Bae. A hybrid approach to feature segmentation of triangle meshes. *Computer-Aided Design*, 35(9):783–789, 2003a.

Anshuman Razdan and Myungsoo Bae. A hybrid approach to feature segmentation of triangle meshes. *Computer-Aided Design*, 35(9):783–789, August 2003b. URL http://dx.doi.org/10.1016/S0010-4485(02)00101-X.

Chris Richardson. ORM in dynamic languages. *Queue*, 6:28–37, May 2008. ISSN 1542-7730. URL http://queue.acm.org/detail.cfm?id=1394140. [Online; accessed 8-Aug-2013].

Helge Ritter. Self-organizing maps on non-euclidean spaces. In *Kohonen Maps*, pages 97–110. Elsevier, 1999.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the Sixth International Conference on*

Computer Vision, ICCV '98, pages 59–, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 81-7319-221-9. URL http://portal.acm.org/citation.cfm?id=938978.939133. [Online; accessed 8-Aug-2013].

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40:99–121, November 2000. ISSN 0920-5691. doi: 10.1023/A:1026543900054. URL http://portal.acm.org/citation.cfm?id=365875.365881. [Online; accessed 8-Aug-2013].

R. Prost S. Valette, J.-M. Chassery. Acvd : Surface mesh coarsening and resampling, 2008. URL http://www.creatis.insa-lyon.fr/site/en/acvd. [Online; accessed 8-Aug-2013].

Firooz A. Sadjadi and Ernest L. Hall. Three-dimensional moment invariants. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-2(2):127 –136, march 1980. ISSN 0162-8828. doi: 10.1109/TPAMI.1980.4766990.

Dietmar Saupe and Dejan V. Vranic. 3d model retrieval with spherical harmonics and moments. In B. Radig and S. Florcyzk, editors, *23rd DAGM Symposium*, volume 2191/2001 of *Lecture Notes in Computer Science*, pages 392–397, Munich, Germany, 2001. Springer-Verlag. 2191.

Will Schroeder, Kenneth M. Martin, and William E. Lorensen. *The Visualization Toolkit, An Object-Oriented Approach To 3D Graphics, 4th edition*. Kitware, U.S.A., 2006. ISBN 193093419X.

P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser. The princeton shape benchmark. In *Proceedings of International Conference on Shape Modelling and Applications 2004*, pages 167–178, Genove, Italy, 2004. IEEE.

Oana Sidi, Oliver van Kaick, Yanir Kleiman, Hao Zhang, and Daniel Cohen-Or. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Transactions on Graphics, (Proceedings SIGGRAPH Asia 2011)*, 30(6):126:1–126:9, 2011.

James Stewart. *Calculus: Fourth Edition*. Brooks/Cole Publishing Company, California, USA, 1999. ISBN 0-534-35949-3.

R. Sucher. A self-organising nonlinear noise filtering scheme. In *Signals, Systems and Computers, 1995. 1995 Conference Record of the Twenty-Ninth Asilomar Conference on*, volume 1, pages 681 –684 vol.1, 30 1995-nov. 1 1995. doi: 10.1109/ACSSC.1995.540636.

H. Sundar, D. Silver, N. Gagvani, and S. Dickinson. Skeleton based shape matching and retrieval. In *Shape Modeling International, 2003*, pages 130 – 139, may 2003. doi: 10.1109/SMI.2003.1199609.

R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, USA, March 1998. ISBN 0-262-19398-1.

Hung-Khoon Tan and Chong-Wah Ngo. Localized matching using earth mover's distance towards discovery of common patterns from small image samples. *Image Vision Comput.*, 27:1470–1483, September 2009. ISSN 0262-8856. URL `http://portal.acm.org/citation.cfm?id=1555006.1555127`. [Online; accessed 8-Aug-2013].

J. W. H. Tangelder and R. C. Veltkamp. A survey of content based 3d shape retrieval methods. In *Shape Modeling Applications, 2004. Proceedings*, pages 145–156, 2004. URL `http://dx.doi.org/10.1109/SMI.2004.1314502`. [Online; accessed 8-Aug-2013].

Francois Preteux Titus Zaharia. New content for the 3d shape core experiment the 3d cafe: data set, March 2000.

R. Toldo, U. Castellani, and A. Fusiello. The bag of words approach for retrieval and categorization of 3d objects. *The Visual Computer*, 26(10):1257–1268, October 2010.

Tony Tung and Francis Schmitt. The augmented multiresolution reeb graph approach for content-based retrieval of 3D shapes. *International Journal of Shape Modeling*, 11:91–120, 2005. doi: 10.1142/S0218654305000748.

S. Valette, J.-M. Chassery, and R. Prost. Generic remeshing of 3d triangular meshes with metric-dependent discrete voronoi diagrams. *Visualization and Computer Graphics, IEEE Transactions on*, 14(2):369 –381, march-april 2008. ISSN 1077-2626. doi: 10.1109/TVCG.2007.70430.

Jean-Philippe Vandeborre, Vincent Couillet, and Mohamed Daoudi. A practical approach for 3d model indexing by combining local and global invariants. In *3DPVT*, pages 644–647. IEEE Computer Society, 2002. ISBN 0-7695-1521-5.

J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas. Self-organizing map in matlab: the som toolbox. In *Proc. of Matlab DSP Conference 1999, Espoo, Finland, November 16–17*, pages 35–40, 1999a.

Juha Vesanto. Som-based data visualization methods. *Intell. Data Anal.*, 3(2):111–126, 1999.

Juha Vesanto, Johan Himberg, Esa Alhoniemi, and Juha Parhankangas. Self-organizing map in matlab: the som toolbox. In *In Proceedings of the Matlab DSP Conference*, pages 35–40, 1999b.

Miguel Vieira and Kenji Shimada. Surface mesh segmentation and smooth surface extraction through region growing. *Computer Aided Geometric Design*, 22(8):771–792, November 2005. URL `http://dx.doi.org/10.1016/j.cagd.2005.03.006`. [Online; accessed 8-Aug-2013].

A. Vincent, R. Christian, and H. Fabrice. Spatio-temporal segmentation using 3d morphological tools. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 3, pages 877 –880 vol.3, 2000. doi: 10.1109/ICPR.2000.903683.

Domenico Visintini, Fabio Crosilla, and Francesco Sepic. Laser scanning survey of the aquileia basilica (italy) and automatic modeling of the volumetric primitives. In *Archives of IAPRS Volume XXXVI, Part 5*, 2006.

Ellen M. Voorhees and Lori P. Buckland, editors. *Proceedings of the Fifteenth Text REtrieval Conference, TREC 2006, Gaithersburg, Maryland, November 14-17, 2006*, volume Special Publication 500-272, 2006. National Institute of Standards and Technology (NIST).

D.V. Vranic and D. Saupe. Description of 3d-shape using a complex function on the sphere. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, volume 1, pages 177 – 180 vol.1, 2002. doi: 10.1109/ICME.2002.1035747.

D.V. Vranic, D. Saupe, and J. Richter. Tools for 3d-object retrieval: Karhunen-loeve transform and spherical harmonics. In *Multimedia Signal Processing, 2001 IEEE Fourth Workshop on*, pages 293 –298, 2001. doi: 10.1109/MMSP.2001.962749.

Xiaojun Wan and Yuxin Peng. The earth mover's distance as a semantic measure for document similarity. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 301–302, New York, NY, USA, 2005. ACM. ISBN 1-59593-140-6. URL `http://dl.acm.org/citation.cfm?id=1099637`. [Online; accessed 8-Aug-2013].

Eric W. Weisstein. Normal curvature, 2012a. URL `http://mathworld.wolfram.com/NormalCurvature.html`. [Online; accessed 12-Dec-2012].

Eric W. Weisstein. Sigmoid function, 2012b. URL `http://mathworld.wolfram.com/SigmoidFunction.html`. [Online; accessed 12-Dec-2012].

Wikipedia. Logistic curve, 2008. URL `http://en.wikipedia.org/wiki/Logistic_function`. [Online; accessed 23-June-2013].

Haim J. Wolfson and Isidore Rigoutsos. Geometric hashing: An overview. *IEEE Comput. Sci. Eng.*, 4:10–21, October 1997. ISSN 1070-9924. URL `http://dx.doi.org/10.1109/99.641604`. [Online; accessed 8-Aug-2013].

Yingxin Wu and Masahiro Takatsuka. Spherical self-organizing map using efficient indexed geodesic data structure. *Neural Networks*, 19(6-7):900–910, 2006.

Z. Wu, Q. Ke, J. Sun, and H.Y. Shum. A multi-sample, multi-tree approach to bag-of-words image representation for image retrieval. In *International Conference on Computer Vision*, pages 1992–1999, 2009.

Lu Xia, Chia-Chih Chen, and J.K. Aggarwal. Human detection using depth information by kinect. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 15 –22, june 2011. doi: 10.1109/CVPRW.2011.5981811.

K. C. Yao, M. Mignotte, C. Collet, P. Galerne, and G. Burel. Unsupervised segmentation

using a self-organizing map and a noise model estimation in sonar imagery. *Pattern Recognition*, 33(9):1575–1584, 2000.

N. Yokoya and M.D. Levine. Range image segmentation based on differential geometry: a hybrid approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(6): 643–649, 1989. TY - JOUR.

Cha Zhang and Tsuhan Chen. Efficient feature extraction for 2d/3d objects in mesh representation. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 3, pages 935 –938 vol.3, 2001. doi: 10.1109/ICIP.2001.958278.

Dongmei Zhang and M. Hebert. Harmonic maps and their applications in surface matching. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages 2 vol. (xxiii+637+663), 1999. doi: 10.1109/CVPR.1999.784731.

Xing Zhang, Jie Tian, Kexin Deng, Yongfang Wu, and Xiuli Li. Automatic liver segmentation using a statistical shape model with optimal surface detection. *Biomedical Engineering, IEEE Transactions on*, 57(10):2622 –2626, oct. 2010. ISSN 0018-9294. doi: 10.1109/TBME.2010.2056369.

Qing-Fang Zheng and Wen Gao. Constructing visual phrases for effective and efficient object-based image retrieval. *ACM Trans. Multimedia Comput. Commun. Appl.*, 5:7:1–7:19, October 2008. ISSN 1551-6857. URL `http://doi.acm.org/10.1145/1404880.1404887`. [Online; accessed 8-Aug-2013].